Master's thesis
Master's Programme in Data Science

Transfer Learning with Label Adaptation for Counterparty Rating Prediction

Tlahui Bolaños

May 26, 2021

Supervisor(s): Assistant Professor Arto Klami
David Bolder, Head of Model Development
at Nordic Investment Bank

Examiner(s): Assistant Professor Arto Klami
Assistant Professor Luigi Acerbi

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Tlahui Bolaños

Työn nimi — Arbetets titel — Title

Transfer Learning with Label Adaptation for Counterparty Rating Prediction

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	May 26, 2021	51

Tiivistelmä — Referat — Abstract

Credit rating is one of the core tools for risk management within financial firms. Ratings are usually provided by specialized agencies which perform an overall study and diagnosis on a given firm's financial health. Dealing with unrated entities is a common problem, as several risk models rely on the ratings' completeness, and agencies can not realistically rate every existing company. To solve this, credit rating prediction has been widely studied in academia. However, research in this topic tends to separate models amongst the different rating agencies due to the difference in both rating scales and composition. This work uses transfer learning, via label adaptation, to increase the number of samples for feature selection, and appends these adapted labels as an additional feature to improve the predictive power and stability of previously proposed methods. Accuracy on exact label prediction was improved from 0.30, in traditional models, up to 0.33 in the transfer learning setting. Furthermore, when measuring accuracy with a tolerance of 3 grade notches, accuracy increased almost 0.10, from 0.87 to 0.96. Overall, transfer learning displayed better out-of-sample generalization.

ACM Computing Classification System (CCS):
Applied computing → Law, social and behavioral sciences → Economics
Computing methodologies → Machine learning → Learning paradigms →
Multi-task learning → Transfer learning

Avainsanat — Nyckelord — Keywords

Credit Ratings, Transfer Learning, Label Adaptation, Feature Selection, Machine Learning

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Somehow, we'll find it. The balance between whom we wish to be and whom we need to be. But for now, we simply have to be satisfied with who we are.

— Brandon Sanderson, The Hero of Ages

Contents

1. Introduction

Good risk management policies can make the difference as to whether a company survives or goes bankrupt in times of crisis. Internal risks are, overall, easier to quantify and manage than external risks, as they usually possess direct flow of information. This has long served as an incentive for specialized entities called Rating Agencies to carefully analyze the holistic condition of various companies to perform a statement regarding firms' creditworthiness. Said statements, in the form of credit ratings, are widely used in finance to assist in measuring counterparty risk. For a great introduction to credit risk modelling, and examples on how these ratings are used in the real world see [1].

In the light of recent events, such as the 2008 global financial crisis, it is obvious that credit ratings are far from perfect. Nevertheless, they still provide a relatively straightforward way of measuring risk. This precedent, along with modifications in the regulations all over the world, such as [2][†], have encouraged financial companies to develop their own recipes for rating counterparties rather than completely relying on an external agency's rating. Furthermore, companies focusing on a really specific niche have trouble finding ratings for their counterparties, which may provide more arguments in favor of developing an internal methodology.

As one might guess, creating a brand new rating methodology is no trivial task. Therefore, it is a popular approach to leverage existing ratings and use them to predict the values of non-rated counterparties using, for example, machine learning.

Machine Learning (ML) is a field of study in computer science that aims to improve performance in a given task by means of experience. In particular, it refers to automating an activity (the task) with a computer (the machine) which usually achieves this by performing some sort of analysis on collected data (learning from experience). It is a concept often thrown alongside Artificial Intelligence (AI) and Deep Learning (DL), and some people may find the distinction confusing. For this study it is sufficient to understand that ML is a branch of AI, with the difference that ML focuses on automation and does not care to dwell or describe what an intelligent

[†]Technically this is not regulation, it is more of a recommendation. However, most of the regulations on banking systems in the world are based on this.

machine is. DL, on the other hand, is a category of ML algorithms. The term DL is used for Neural Networks characterized by the large number of layers (and consequently, large amounts of data) constructed within them.

ML provides, in principle, a fairly easy way to implement a solution for the rating problem. One can gather descriptive data along with the ratings corresponding to various firms, train a model on this data and use it to predict the unobserved cases. It provides automation, prediction and quality control already implemented in many different programming languages (e.g. Python, R, etc.) and tools. Different approaches have been developed for this problem, ranging from simple linear regressions to complex Convolutional Neural Networks and Self-Organizing Maps (for a good survey on several methods, see [3]). However, one detail that has been largely ignored is the fact that real world applications often lack the necessary amount of data.

This statement might even sound polemic in the so called "Era of Information", but a key factor for success in ML is that the data used for training the model has to be equivalent to the data in the prediction. This is the first obstacle for a ML solution within the context of this study: rated counterparties are, in most cases, big known companies from very selective regions, whereas the internal model is usually applied on smaller and more specific firms. Naively training on these counterparties will most likely bias the predictions for these smaller, niche focused companies.

Consequently, Transfer Learning (TL) might prove useful to solve this potential issue. TL is a ML concept, which combines the knowledge obtained from a well known task and uses it to either get a head start or to improve performance on a new related problem. Let us imagine, for example, that we intend to build a Neural Network to recognize images of trucks. Suppose further that we already created and trained a Neural Network for recognizing cars in a previous study. The two tasks are not identical, but they are similar enough that we can use the trained car network as a starting point to accelerate the training on the truck network by inheriting the parameters. This is one example of TL. TL has been used for credit scoring in [4] and [5], but to the best of our knowledge it is yet to be explored in counterparty rating.

In brief, the objective of this study is to implement a TL setting for counterparty rating prediction, which could be applied with minor modifications by an interested firm within their risk management operations either as an internal model, or as a validation tool for their existing rating process. Hence, out-of-sample performance is of the utmost importance, as the introduction of new unobserved counterparties should yield stable results.

The rating information from three of the biggest agencies (Standard & Poor's, Moody's and Fitch) was used to improve a prediction task on the ratings provided by A.M. Best using Label Adaptation. This particular instance of Label Adaptation

consists of mapping all the ratings into a numeric space, and subsequently translating the information into a unique rating, namely the one corresponding to A.M. Best. We propose a new method, Stable Linear Regression, as a way to maintain the interactions between the different agencies when adapting the labels.

After this, we matched the ratings with a financial dataset and implemented feature selection with a Genetic Algorithm on the dataset extended by Label Adaptation. Classification and regression variants of the Multi-Layer Perceptron and the Support Vector Machine were tested with and without the adapted labels as extra features for comparison. The testing was performed using a 5-fold Cross Validation using accuracy as the validation measure. A prediction was considered as accurate if it was between n grade notches of distance to the actual observation, with $n \in \{0, 1, 2, 3\}$. Leveraging Transfer learning, we managed to improve the mean accuracy of the prediction at $n = 0$ from 0.30 to 0.33, and for $n = 3$ from 0.87 to 0.96. Although the improvement seems marginal at first glance, the careful inspection of the confusion matrix provides a better generalization for out-of-sample prediction, making the TL implementation considerably more robust.

The rest of the document is organized as follows:

- **Chapter 2 (Previous work)** provides a summary on techniques for feature space reduction and presents some modelling approaches used with direct ML, as well as related Transfer Learning applications.

- **Chapter 3 (Framework)** explains in detail the experimental setup studied. It starts by defining and exploring the data used, describes the implementation of Transfer Learning within the setup and lists the different individual models to be utilized. Furthermore, it also discusses how the results will be measured compared to ground truth.

- **Chapter 4 (Results)** reports the various findings of the study. It also provides a direct comparison to more direct methods, e.g. applying the individual models.

- **Chapter 5 (Discussion)** states some important facts about the study, possible shortcomings and improvements, as well as different implementations that were considered but not explored.

- **Chapter 6 (Conclusions)** provides a compact summary of the methodology and results.

2. Previous work

Considerable amount of research has been s done in the topic of credit rating prediction, albeit most of it is focused in credit scoring rather than determining counterparty rating as in [4] and [6]. The main difference between these two is that credit scoring is performed on individuals with a numeric grade while credit rating is applied on firms upon a classification scale. As such, both the inputs and the outputs of the process are similar, but not identical. Novel approaches in credit scoring may serve as an inspiration, but one has to be careful as the data available in scoring is orders of magnitude more abundant, which may lead to serious convergence issues. To provide some context, in December 2019 Standard and Poor's reported $1,068,942$ outstanding credit ratings in [7], while Experian states in [8] that it maintains credit information on over 220 million people in 2021 just within the U.S.

The literature that does indeed refer to counterparty rating usually utilizes the ratings as labels matched with each firm's own financial information to construct a classification problem[†]. Standard procedure is undertaken for data preparation, which includes, among others, normalization, missing data imputation, matching of different datasets, etc. This step of the process is often implied but not thoroughly discussed, and can be avoided by collecting pre-processed data.

The main task in this study can be described as follows. Given a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ where N is the number of observations and D is the number of features or explanatory variables, and a vector of ratings $\boldsymbol{y} \in \mathbb{R}^N$, the goal is to learn a function $f : \boldsymbol{X} \to \boldsymbol{y}$. A large D can cause issues for training models, as it often includes high correlations and can lead to over-fitting if it is relatively close to the size of N. Therefore, two clear steps are described in the predictive process: reducing the amount of features and developing a model framework. The former aims both to reduce over-fitting and, in some cases, also to deal with model complexity and interpretability. The latter is responsible for the actual prediction of the rating.

The two steps in the modelling process will be further discussed in Sections 2.1 and 2.2, where brief descriptions of different techniques and their respective results will be provided. Validation for the modelling choices will be examined by Section

[†]Again in reference to [3], most studies cited in this paper present the mentioned structure.

2.2.2. The final topic for this chapter is Transfer Learning. Although the studies in TL related to counterparty rating are few and far between, it is easier to adapt for seemingly unrelated topics as it involves model structure. Classifications have been provided in [9] and [10], which proved quite useful in this endeavor. Section 2.3 will lay out an introduction to the concepts and definitions studied by TL.

2.1 Dimensionality reduction

One might be tempted to gather all the data available, within reason, and use it to train the ML model. However, two main issues will inevitably prevent us from doing so. First is the fact that there are literally hundreds of different explanatory features in finance, providing the ability to construct new meaningful combinations of features as one so desires. This makes the process of training much slower and potentially unfeasible for a real world application. The second issue relates to feature redundancy and noise. It has been demonstrated that carefully selecting the features in a model can improve performance in both computation time and error minimization, as shown by the results cited in Sections 2.1.1 and 2.1.2.

The general consensus divides feature reduction into feature selection, in which the variables that better explain the results are chosen given some ranking or scoring algorithm; and feature projection, which aims to provide a low-dimensional representation of the feature space. Feature selection maintains interpretability while feature projection tends to boost overall accuracy.

Feature selection algorithms can be categorized, for example, as in [3]:

- **Filters:** They measure explanatory power from the data independently from the model. The features that best represent the labels are separated from the rest.
- **Embedded Feature Selection:** In this category, feature importance is measured within the context of the model. This usually results in a more tailored approach, which might turn computationally expensive to implement for a broad selection of models.
- **Wrappers:** To explain wrappers we must first understand the underlying brute force method. It consists on exploring the entire power set $\mathcal{P}(X)$ of features, and measuring their effectiveness. This guarantees that we know how different subsets compare in predictive potential, but can turn quite troublesome for large feature spaces. Wrappers here provide a way to systematically navigate this potentially enormous power set to limit the search and find a satisfactory reduction with realistic time and resources constraints.

Additionally, feature projection is often divided by linear and non-linear models. In a series of test performed in [11], it was concluded that non-linear feature projection has good performance on artificial tests, but still lags behind linear variants such as Principal Component Analysis (PCA) for real-world problems. It is worth noting that ever since those tests were carried out in 2009, variants of the non-linear algorithms have been designed to deal with their intrinsic weaknesses in real-world applications, as will be seen in Section 2.1.2.

2.1.1 Feature selection

Feature selection is defined, for the purpose of this study, as the careful examination and adoption of a subset of the original feature space. This subset will then be utilized as the data on which the subsequent models in the framework will be trained.

We have already discussed the technical reasons that encourage the reduction of the feature space for rating predictions. However, there is another equally important reason for companies to perform feature selection in particular: financial information is expensive. Official information regarding this is quite elusive, as prices are not usually stated upfront on the websites. As an example, a simple internet search on Bloomberg provides prices ranging from $20,000 - $25,000 USD annually per terminal, which is the physical access point for the information. Migrating this information to other computers or systems causes further expenses, as this can not be done with just the terminal. Financial information can provide an important edge for companies, but it also translates into elevated overall costs, as the previous example illustrates. As such, carefully selecting a subset of the feature space can, more often than not, provide substantial savings.

Feature selection techniques also naturally retain the intrinsic interpretability of the models. This is because we retain the original features, and having lower dimensions makes it easier to convey visual representations. On the other hand, feature projection often struggles in this area, as projections construct new features in a complex fashion. These new projected features are often difficult to interpret, although again easier to visualize.

Next, we will explore filters, embeddings and wrappers applied for selecting features in counterparty rating prediction. The discussion will also introduce a handful of actual ML models. We ask the reader to be patient if all the acronyms start to sound confusing. Section 2.2 will hopefully clarify the main characteristics of each one.

Filters

Filters, as explained in [3], select a subset of features by implementing an evaluation function. This function provides an estimate of the usefulness of each feature and thus provides a way to separate them.

Due to their lower computational power requirements, filters were amongst the first feature selection techniques to be implemented in literature. Analysis of Variance (ANOVA), in particular, was used in [12] to compare the performance of Support Vector Machines (SVMs) and Multi Layered Perceptron (MLP). This type of filtering is used to compare whether the means of two or more populations are equal. They selected the features using the p-values extracted from the ANOVA. In [12], data from companies of US and Taiwan along with their corresponding Standard and Poor's ratings was used for the comparison. The authors found that the feature selection improved the accuracy of the models for most of the configurations, by reducing from 19 to 14 features in the Taiwan case, and 17 to 12 features in the US case.

Independent-samples t-test and stepwise discriminant analysis were later used in [13] along with an improved version of SVMs, namely, the Ordinal Multi-class Support Vector Machines (OMSVMs). Details of the model will be discussed in 2.2. The independent-samples t-test is similar to ANOVA, but it can only compare the means on a pairwise basis. The discriminant analysis was performed using Wilk's lambda, which is an extension on ANOVA. No comparison was provided between the filtered and unfiltered performance, but the method reduced the data from 39 to 14 features.

Embedded feature selection

Embedded feature selection is implemented in a similar way as in filters, with the main difference that the evaluation function is linked to a prediction model. This means that the feature selection step must be performed for each individual model to be tested.

Perhaps the clearest example of this type of selection is the adaptation of Linear Regression, namely the least absolute shrinkage and selection operator (LASSO) regression. This type of regression trains from the data and scores the predictive power of the features at the same time by adding an extra term to the optimization problem. By doing so, it can measure each feature's individual contribution. It was employed in [14] to reduce the number of features from 268 to between 45 - 95, depending on the configuration. This resulted in an overall improvement in the out-of-sample prediction, more than doubling the accuracy for some cases.

Neighborhood rough set theory was leveraged in [15] using a Particle Swarm Optimization (PSO) algorithm to tune the parameters. The new embedded method is called Rough-Set-based Feature Selection Algorithm for Imbalanced Data (RSFSAID).

It takes into account the different distributions of the labels to produce the selection. It first uses neighborhood rough set theory to calculate boundary regions between the variables. These boundary regions describe where any given model could potentially misclassify for the objective variable. This gives three parameters, which are later optimized by PSO based on the model to be used. [15] tested the method in various different study cases, although none of them is related to credit rating. It is still relevant to our problem as it takes into account the imbalanced data, which is quite common in ratings. RSFSAID reduced the amount of features to an average of about 40% from the original feature space size while improving the area under the ROC* curve (AUC) an average of 7% for a specific Decision Tree algorithm implemented in Java, 1% for Random Forests (RFs), 4% for NaiveBayes and 27% for SVMs†.

Wrappers & Hybrid methods

While filters and embeddings can yield good results on feature selection, they also tend to ignore the interactions between features. There have been efforts to extend these algorithms to encompass pairwise relationships between variables, but to completely explore all the possibilities one has to work with the power set $\mathcal{P}(X)$ of features. Wrappers provide a solution for exploring the combinations of features in a smart way. Even when leveraging wrappers, however, the complexity of the computations is extremely sensible to the number of features and the size of the dataset. This is why many studies have combined wrappers with filters and/or embeddings to simplify and accelerate feature selection.

The Information Gain Directed Feature Selection algorithm (IGDFS) was used in [16] alongside k-Nearest Neighbors (k-NN), NaiveBayes and SVMs. This method uses Information Gain (IG), which measures the reduction in entropy induced by a feature, as an heuristic within a Genetic Algorithm Wrapper (GAW) for exploring the feature power set. Then it evaluates the accuracy of the feature subset using the classifiers. There are two main characteristics that the classifiers must satisfy for this approach: they must be able to operate over large feature spaces, as the GAW may select a large subset to run tests on; and they must be able to provide a representation of the results in term of probabilities for the IG. The main results in [16] showed that, again, feature selection improved performance of the models, with IGDFS having a slight edge over plain GAW.

Three main requirements for feature selection are described in [17]: The subset selected should provide high performance, it should be robust to imbalanced data and

*ROC refers to the Receiver Operating Characteristic curve, see Section 2.2.2.

†This large improvement is a bit misleading, because SMVs were the worst performing model for the original features by a large margin. However, the general idea is that there is, in fact, improvement.

it should have a manageable computational complexity. As we can observe, IGDFS addresses only the first and last requirements. Thus, [17] also provides the Ensemble Learning-based Filter-centric hybrid feature selection (ELFS) as an alternative to cover all three requirements. It works by combining subsampling to deal with imbalance, filtering to get rid of irrelevant features, and embedding models (and their pruning) to provide a score on each feature. ELFS was tested on 8 different datasets, and compared to 4 other hybrid feature selection techniques. The authors found a trade-off between the performance and the complexity of the computations. This means that we may want to use ELFS for exceptionally big datasets, but for smaller ones one might prefer some better performing hybrid method.

2.1.2 Feature projection

As we have already mentioned, this family of dimensionality reduction projects the feature space into a lower dimension, while trying to maintain the greatest amount of information possible. This approach often out-performs feature selection and usually with a lower computational complexity. This might be because dropping features may introduce a greater information loss than projecting them. However, interpretation is often lost in this setting, which is peculiarly problematic for a real-world financial implementation.

Some structural work-arounds have been developed to deal with this issue. A popular approach is to group the features by what area of the company they are describing, and then perform feature projection on each group individually. By doing this, we might lose track of what the new features exactly represent, but we retain an overall idea of what they mean. For example, most rating agencies (and stemming from them, also a lot of research in the area) distinguishes two main risk profiles for features: business and financial. The business profile deals mostly with external variables such as the location and industry of the company. The financial profile dwells with the inner workings of the company, such as leverage, liquidity, solvency, capital adequacy etc.

Within feature projection, methods can be categorized by the type of projection used as linear and non-linear. Principal Component Analysis (PCA) is the most widely used technique due to its overall good performance and low computational cost. It is also quite useful for data visualization, as are most of feature projection techniques. As such, many of the techniques that we will describe are directly compared to PCA's performance.

Linear methods

We will start by providing a definition on linear feature projection as in [18], which is quite simple and provides a good overview of the concept:

Definition 2.1.1 (Linear Feature Projection)**.** Given a dataset $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d] \in \mathbb{R}^{n \times d}$, where n is the number of samples and d is the number of features, feature projection is the optimization of an objective function $f_{\boldsymbol{X}}(\cdot)$ to produce a linear transformation $\boldsymbol{P} \in \mathbb{R}^{d \times r}$, $r < d$. The resulting representation $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{P} \in \mathbb{R}^{n \times r}$ is called the low-dimensional transformed data.

The main difference between each linear technique is the choice of $f_{\boldsymbol{X}}(\cdot)$. PCA utilizes the maximization of $f_{\boldsymbol{X}}(\boldsymbol{P}) = \text{trace}\left(\boldsymbol{P}\text{cov}(\boldsymbol{X})\boldsymbol{P}^T\right)$. In other words, it tries to conserve the most variance possible while reducing the dimensionality. In [19] PCA was compared to a double-bounded tree-connected isometric feature mapping (dbt-Isomap). The study comprised short and long term ratings from Fitch for various banks divided in three main regions: European Union, United States and Asia. They utilized 23 features, which were reduced for EU, US and Asia, respectively, into $(5, 4, 5)$ for PCA and $(6, 4, 5)$ for dbt-Isomap. PCA performed better overall than the non-reduced setting with dbt-Isomap being the best performing alternative.

Non-linear methods

Naively implemented non-linear methods suffer in real-world applications as stated by [11]. In this study, many non-linear techniques were compared to PCA in both artificial and real-world datasets. None of the techniques could consistently out-perform PCA in the real-world experiments. The authors describe three main weaknesses that might explain this result: underlying properties of the objective functions could be heavily skewing the results towards trivial solutions, some of the optimization processes have been found to be sensitive to numerical issues, and they have greater susceptibility to the curse of dimensionality due to non-linearity.

We begin with dbt-Isomap as we have mentioned it previously. Isomaps work, as explained in [20], by building a graph using the k-nearest neighbors or the points within a hyper-sphere of radius ε. It performs the projection by optimizing on the geodesic distance between the nodes, where the weight of the vertices is the euclidean distance between points. This comes with many issues, as the graph must be fully connected for this to work, which usually causes instability due to short-circuits as noted in [11]. Because of this, [19] implemented a variation of this algorithm, the dbt-Isomap. This improvement starts by building a graph by connecting at most k-nearest neighbors within a radius ε for each point. Then, for each connected component, it calculates

a centroid, selects the point closest to that centroid, called the approximate centroid, and builds the minimum spanning tree amongst the approximate centroids to build the fully connected graph. Their results show that this method of building the graph performs better than PCA for this real-world application.

Kernel Graph Embedding (KGE) was compared to PCA and Recursive Feature Elimination (RFE) in [21] for credit rating. As explained by [11], KGE works by implementing a non-linear kernel within the linear graph embedding algorithm for feature projection. RFE, on the other hand, works by eliminating the worst performing features iteratively. The study was performed using 38 features from various companies in Taiwan. The number of resulting features was set to 5, and KGE was the best performing algorithm of the group due to the non-linearity of the underlying data.

2.2 Modelling

Once the feature space has been reduced, we must transition into the prediction phase of the process. ML has seen a lot of growth in the latest years, with many stand-alone models and improvements on existing approaches developed in a short period time. Studies often explore the possibility of combining several of these models to improve performance. These combinations, often called ensembles (see [22] for a more detailed explanation), can be arranged into three main categories:

- **Bagging:** This structure trains several models in parallel, and then unifies the results using some overall measure, such as the average.
- **Boosting:** Individual models are trained sequentially, where the outputs of a model are the inputs for the following model. In the end all results are aggregated into a single output.
- **Stacking:** Similarly to bagging, the models are trained in parallel. However, this technique leverages a master model, which combines the results into a single output.

As discussed in [22], one has to be careful when dealing with ensembles, as they can lead to over-fitting issues. Naive boosting, as it is trained in a sequence, is particularly prone to this issue when using a large amount of individual learners. However, bagging and stacking are often more resilient. When implemented correctly, ensembles are able to extract the from properties from many individual models. Just mentioning all the different ensembles that have been utilized in the credit rating prediction problem would constitute a rather lengthy survey, as anyone can construct their own

personalized configuration, e.g. [23], [24] and [25]. As such, we will limit our attention to a few of the most important individual models and mention some notorious ensemble implementations and results.

2.2.1 Individual models

Neural Networks (NNs)

NNs are inspired by the way brains work in real life. As described by [26], they are composed of individual processors called neurons. Each neuron receives a set of N_j inputs, which it combines using an activation function $f(\cdot)$ as follows:

$$y = f\left(\sum_{i=1}^{N_j} w_i x_i\right).$$

The incoming signals or inputs are weighted within the activation function, and then passed on to subsequent neurons. The connections between each of the neurons define a connected graph, with the w_i being the weights assigned to the edges coming from a given node, which in turn represents a neuron. Depending on the structure of the aforementioned graph, one can build a NN for many different purposes.

The most commonly used NN for classification purposes is the Multi-Layer Perceptron (MLP). The graph of an MLP is a Directed Acyclic Graph (DAG). This ultimately means that the information flows in one direction without forming any loops, resulting in a structure composed by various layers of neurons as shown by Figure 2.1. The first layer connects directly to the data, and is called the input layer. The output layer, which comes last in the graph, provides the results of the classification. All layers in between are referred to as hidden layers. MLPs are usually trained using Stochastic Gradient Descent (SGD) implemented by a backpropagation algorithm. If the number of layers is sufficiently large, the MLP is classified as a deep neural network which, according to [27], have become really popular and won a variety of ML competitions as of late.

Another common implementation of NNs are Self-Organizing Maps (SOMs), as introduced by [28]. For this, neurons are organized in a grid structure, where each node contains an vector in the original n-dimensional feature space of the data. The network is then trained in an unsupervised manner using, where each data point affects the neuron it most closely resembles, as well as its direct neighbors. This method is a good tool for visualization, as it conserves the topological space, as well as clustering and dimensionality reduction.

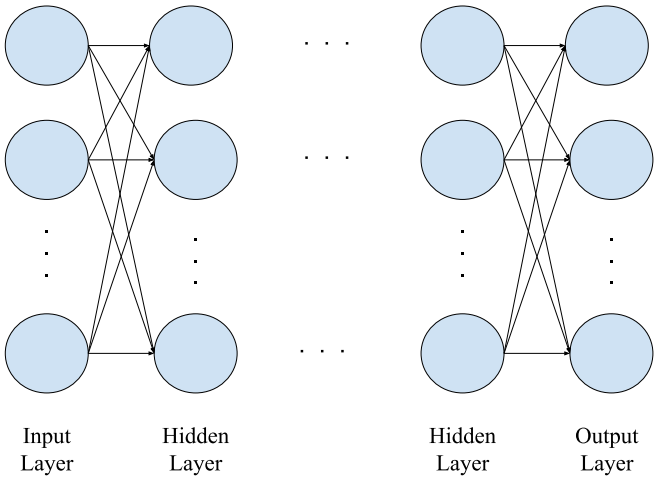

Figure 2.1: Generic structure of a MLP.

Support Vector Machines (SVMs)

SVMs were originally created in [29] as binary linear classifiers for linearly separable data. The idea is to find the hyperplane $\boldsymbol{w}^T\boldsymbol{x} - b = 0$ that separates the two classes by maximizing the distance from the points to the plane. This effectively creates two margin hyperplanes $\boldsymbol{w}^T\boldsymbol{x} - b = \pm 1$, where the distance between the margins is $2/\left\|\boldsymbol{w}\right\|$. As the data is linearly separable, no point crosses the margin for the two classes, with the closest points to the hyperplane actually laying on the margins. If the labels are set as $y_i = \pm 1$, then the constraints on the margins can be expressed as $y_i(\boldsymbol{w}^T\boldsymbol{x} - b) > 1$. Thus, the optimization is done by minimizing $\|\boldsymbol{w}\|$ within the aforementioned constraints, which turns out only to depend on the subset of points laying on the margin, which are in turn called support vectors.

This method is only applicable with a hard margin that completely separates the classes. To extend it to more applications, a soft margin was defined with the hinge loss as $\max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x} - b))$. This loss function only penalizes the points lying on the wrong side of the margin. Adding this to the optimization, we now minimize

$$\underset{\boldsymbol{w}}{\text{minimize}} \left(\left[\frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x} - b)) \right] + \lambda \left\|\boldsymbol{w}\right\|^2 \right). \tag{2.1}$$

Equation 2.1 can be modified with a non-linear kernel K to solve more complex problems. It has also been extended to deal with multi-class labels by dividing the original problem into binary sub-problems, which is the case for [12] and [13].

Table 2.1: Confusion Matrix

		Predicted	
		True	False
Observed	True	True Positive (TP)	False Negative or Type II errors (FN)
	False	False Positive or Type I errors (FP)	True Negative (TN)

Rule-Based Systems (RBSs)

In this type of models, decision rules are inferred in an automated fashion from the data. This rules, as explained in [30], usually have a conditional form, that is, *"If this happens, then predict that"*. This models are quite popular in the financial sector due to the interpretability of the results. They are easy to visualize in their basic form. The choice of algorithm that creates the rules is what differs from model to model.

One of the most utilized RBS models, albeit it is more of an ensemble than a model, are Random Forests (RFs). Presented in [31], they are created by computing several Decision Trees (DTs) with different parameters and providing the mode of the results. An individual DT is a RBS, where the algorithm for creating rules works from top to bottom, by splitting the dataset on a given feature by optimizing an impurity measure. The impurity measure describes the success of the split in discriminating the different label classes.

2.2.2 Validation and quality measures

Once predictions are performed, we have to test the results of our process. As this is a classification problem, most of the literature validates the results with measures based on the so-called Confusion Matrix. This matrix, as shown in Table 2.1, was originally developed for binary problems. It produces a nice summary of how the predictions are distributed against actual observations. The matrix as such can be extended to encompass all possible combinations in a multi-class setting. This proves to be quite useful in the credit rating as the Confusion Matrix can retain ordinality.

One of the most commonly used metrics that can be derived from the Confusion Matrix is Accuracy. It is calculated as $(\mathrm{TP}+\mathrm{TN})/n$, where n is the number of samples. Accuracy can be easily extended for multi-class problems by using the form CP/n, where CP stands for the number of correct predictions from the model. Given that our

problem concerns ordered classes, CP can be tuned to integrate predictions with a k notch difference from the actual observations, classifying near predictions as correct.

According to [3], type I and type II errors are also commonly used for validation. In the multi-class setting, they define type I error as the false acceptance of a class, calculated by using the weighted average of all the different classes. Type II error was, on the other hand, the false rejection of the correct class, calculated using $1-\text{Accuracy}$.

Finally, another useful tool for visually validating results is the Receiver Operating Characteristic (ROC) curve. It works by plotting the True Positive Rate (TPR) defined as TP/(TP+FN); against the False Positive Rate (FPR) defined as FP/(FP+TN) with various decision thresholds. The idea behind this is that an uninformed guess should result in roughly the same TPR and FPR, which makes the curve a straight line. Thus, the better a model performs, the more the curve tends to the upper left corner of the plot, which in turn means the TPR grows much faster than the FPR. To translate this visualization into a numeric metric, [15] used the Area Under the ROC Curve (AUC) as the metric to evaluate the competing models.

2.3 Transfer learning

In concise terms, transfer learning aims to leverage knowledge obtained from models trained within similar experiments and use it for the objective at hand. Formally, let us define it as in [9].

Definition 2.3.1 (Domain)**.** A domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ is formed by a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ where $X = \{x_1, x_2, \ldots, x_n\} \in \mathcal{X}$.

Definition 2.3.2 (Task)**.** Given a specific domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$ is formed by a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot) : \mathcal{X} \to \mathcal{Y}$.

Remark. Within this context, $f(\cdot)$ is usually approximated from pairs $\{x_i, y_i\}$, with $x_i \in X$ and $y_i \in \mathcal{Y}$. Furthermore, most literature refers to $f(\cdot)$ as $P(y|x)$, although this defines a basis for the classification more so than a mapping. The term $P(y|x)$ will be used throughout this document.

Definition 2.3.3 (Transfer Learning)**.** Given a source domain $\mathcal{D}_S$ with a learning task $\mathcal{T}_S$ and a target domain $\mathcal{D}_T$ with its corresponding task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target function $P_T(x|y)$ in $\mathcal{D}_T$ by leveraging the knowledge contained in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$

The keen observer may realize that $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$ implies having one or more of the following: $\mathcal{X}_S \neq \mathcal{X}_T$, $P_S(X) \neq P_T(X)$, $\mathcal{Y}_S \neq \mathcal{Y}_T$ and $P_S(x|y) \neq P_T(x|y)$. A

classification is proposed in [10] where *Homogeneous Transfer Learning* deals with the case $\mathcal{X}_S = \mathcal{X}_T$ and *Heterogeneous Transfer Learning* deals with the case $\mathcal{X}_S \neq \mathcal{X}_T$.

Basically, TL involves the improvement of a particular task using information obtained from other similar, but not identical, task. The definition presented here does not impose any kind of process or guidelines for using the source information, and the overall idea can be observed in Figure 2.2. This has resulted in a myriad of different approaches throughout the years, which is also why the classifications have to be updated to better encompass the newer methods.

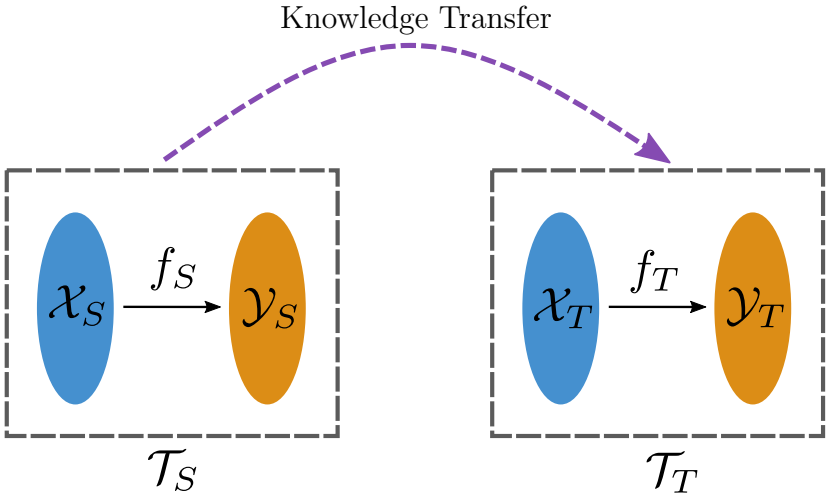

Figure 2.2: Generic structure for Transfer Learning.

To illustrate this better, we refer to some practical examples. In [5], a SOM was introduced to form clusters within the credit scoring dataset. Then, the cluster information was added to the original dataset and modelled with a MLP. This experiment is case of Homogenous TL, where the task differs (clustering from prediction). TL improved the prediction performance for instances were fewer training records were available. Another use case is presented in [4], where they used publicly available data to complement the training phase of a credit scoring process. The main idea is to train a deep neural network on the source (publicly available data), to subsequently retrain segments of the network using the target data. Again, TL showed more improvement the smaller the target data.

3. Framework

The main idea of this project is to investigate whether TL helps in the counterparty rating prediction problem. There are some clear steps for the methodology: retrieving the data (Section 3.1), reducing its the feature space (Section 3.5), constructing a Transfer Learning structure (Section 3.2) and selecting the Machine Learning models to implement and for the knowledge to be transferred (Section 3.3).

The dataset was selected in such a way as to mimic a real world application. The three main rating agencies were selected as source tasks, and a single highly-specialized agency was selected as the target task. This ensures that the distribution and scale of ratings will differ vastly, as described in the following sections.

For the transfer learning portion of the study, the high degree of linearity between ratings was leveraged to perform Label Adaptation, an area of Domain Adaptation discussed in [32]. LA refers to the process of mapping the labels of a different source into a unique space, hence making them more comparable and enabling the transfer of knowledge between different rating agencies and scales. By doing this, we can artificially enlarge our training dataset and hopefully produce more robust results. The transformation from a classification to a regression problem allows us to exploit the linearity and ordinality of the ratings more easily.

Finally, two main ML models, Neural Networks and Support Vector Systems, are proposed for the actual rating prediction. This will allow us to compare different types of results in order to get a better grasp of performance, as they have both classification and regression alternatives.

3.1 Data set

Following the theory described in Section 2.3, there are two main categories of data we need to cover: $\mathcal{X}$ and $\mathcal{Y}$. $\mathcal{X}$, which refers to the features available for the individual firms, is in theory quite straightforward, as we assume that both the target and the source feature spaces coincide ($\mathcal{X}_T = \mathcal{X}_{S_1} = \cdots = \mathcal{X}_{S_J}$). On the other hand, $\mathcal{Y}$, which refers to the ratings given by the agencies for each firm, is similar but not identical within the source and target label spaces. This is due to the fact that each rating

agency has its very own methodology and scale.

However, $\mathcal{X}$ remains simple only in theory, as the different features used to describe the labels are mostly comprised of financial information which has a long history of being difficult and expensive to obtain in practice. This data was obtained from Quandl [33], which has information on around 14,387 different companies. The database spans 2016 to 2021 and includes the features in Table 3.1.

Table 3.1: The 54 features contained in the Quandl database. All features are numerical.

Accounts Payable	EPS - Net Income - Basic	Receivables Turnover
Accounts Payable Turnover	EPS - Net Income - Diluted	Return on Assets
Accrued Expenses Turnover	Free Cash Flow Per Share	Return on Equity
Calculated Tax Rate	Gross Margin	Return on Investment
Cash & Equivalents Turnover	Interest Coverage	Revenue Per Employee
Cash Flow Per Share	Inventories	Revenue Per Share
Cash From Financing	Inventory Turnover	Revenue to Assets
Cash From Investing	Long Term Debt	Shares Outstanding
Cash From Operations	Long Term Debt to Equity	Stockholders' Equity
Cash and Equivalents	Net Current Assets	Total Asset Turnover
Cash, Beginning of Year	Net Income	Total Assets
Cash, End of Year	Net Income Per Employee	Total Assets Per Share
Current Assets	Net Margin	Total Debt to Equity
Current Liabilities	Operating Income	Total Liabilities
Current Ratio	Operating Margin	Total Revenue
Direct Expenses	PP&E Turnover	Weighted Average Shares Outstanding - Basic
EBITDA	Quick Ratio	Weighted Average Shares Outstanding - Diluted
EBITDA Margin	Receivables	Net Assets Per Share

Regarding the label spaces $\mathcal{Y}$, we define the target label space $\mathcal{Y}_T$ as the long term issuer's credit rating from A.M. Best, which is our target domain; while the source spaces $\mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3$ are defined as the long term issuer's credit rating from Standard & Poor's (S&P), Moody's and Fitch Ratings, respectively. The idea behind this configuration is to leverage the label information from the three biggest rating agencies to improve the learning task on a smaller agency. Fortunately, rating agencies are required by law in [34] to make all of their ratings freely available to the public. A.M. Best data was gathered from [35], S&P's data from [36], Moody's data from [37] and Fitch Ratings' data from [38]. The scales for each of the agency's ratings are shown in

Table 3.2.

Table 3.2: The rating scales used by the different agencies, from highest to lowest.

A.M. Best	S&P	Moody's	Fitch	A.M. Best	S&P	Moody's	Fitch
aaa	AAA	Aaa	AAA	⋮	⋮	⋮	⋮
aa+	AA+	Aa1	AA+	bb-	BB-	Ba3	BB-
aa	AA	Aa2	AA	b+	B+	B1	B+
aa-	AA-	Aa3	AA-	b	B	B2	B
a+	A+	A1	A+	b-	B-	B3	B-
a	A	A2	A	ccc+	CCC+	Caa1	CCC+
a-	A-	A3	A-	ccc	CCC	Caa2	CCC
bbb+	BBB+	Baa1	BBB+	ccc-	CCC-	Caa3	CCC-
bbb	BBB	Baa2	BBB	cc	CC	Ca	CC
bbb-	BBB-	Baa3	BBB-	c	C	C	C
bb+	BB+	Ba1	BB+	e	D		D
bb	BB	Ba2	BB	f			
⋮	⋮	⋮	⋮				

As the labels $\mathcal{Y}$ were extracted from a different dataset than the features $\mathcal{X}$, they have to be matched amongst them to get the pairs (x_i, y_i) that serve as input for the modelling process. This proved to be troublesome, as both datasets contained different keys. The ratings were considerably more difficult to treat, as agencies are only expected to fill one of three different keys (CIK, LEI, OI*). To match the individual companies from both data sources, the following defining keys were employed in order:

1. **Central Index Key (CIK):** Unique identifier with the Security and Exchange Commision of the USA. This was the only external key found in the Quandl dataset.

2. **Legal Entity Identifier (LEI):** Unique identifier based on the ISO 17442 standard developed by the International Organization for Standardization. Only the ratings dataset had this key, but there exist services such as [39] which allowed us to convert from LEI to CIK for the matching process.

3. **Exact name:** Companies from both datasets with the same exact name were paired together.

*Obligor Identifier (OI) is internally assigned by each rating agency, rendering it useless for matching purposes

4. **Levenshtein ratio:** A measure of how similar two strings are. It is based on the Levenshtein or edit distance*, and we matched the companies with a ratio greater than 0.95.

In the end, about $2,800$ different companies were matched, which amounts to around $9,700$† accounting various time periods. This was mainly because Quandl's data was heavily focused on the Asian markets, while the agencies have more observations from the North American and European regions.

3.1.1 Exploratory analysis

To confirm that our experiment setting is correct, and to adequately select the methods to utilize, we must first perform an exploratory analysis on the data. With this in mind, we present in Figure 3.1 the number of rated companies per agency. A.M. Best is, by far, the agency with the fewest rated companies. This makes sense, as they focus on the insurance sector, which is a subset of the companies rated by all the other agencies. A.M. Best was selected as the target task precisely for this reason.

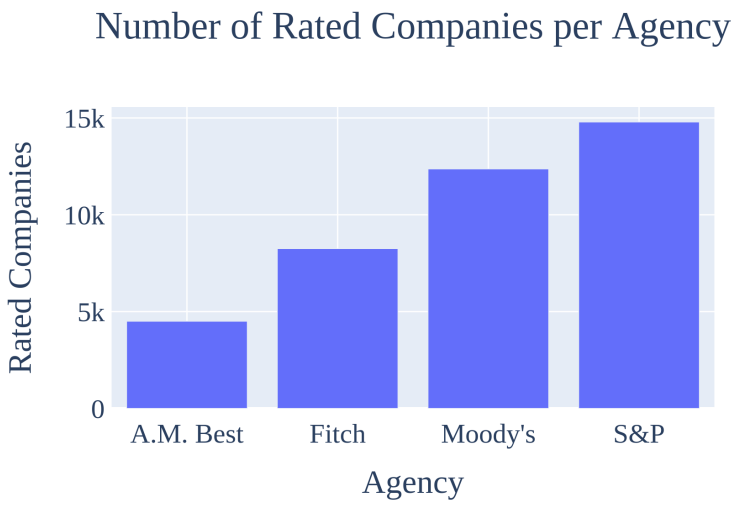

Figure 3.1: Number of rated companies per agency.

As we mentioned before, to perform the cross-comparison of the agencies and to transform the problem into a regression setting, we change the ratings from categorical to numerical data. The transformation is quite simple, we partition the $[0, 1]$ interval into n_r centroid points $C = \{c_1 = 0, c_2, \ldots, c_{n_r} = 1\}$, n_r being the number of different grades on a given rating scale r and $c_i = (i-1)/(n_r - 1)$. Then we perform a mapping

*It measures the minimum amount of changes needed to make the strings equal.

†We have multiple years of rating and financial data for each company, that is why we have more records than companies.

from the rating scale to the centroids, with the lowest grade being mapped to $c_1 = 0$ and the highest grade to $c_{n_r} = 1$.

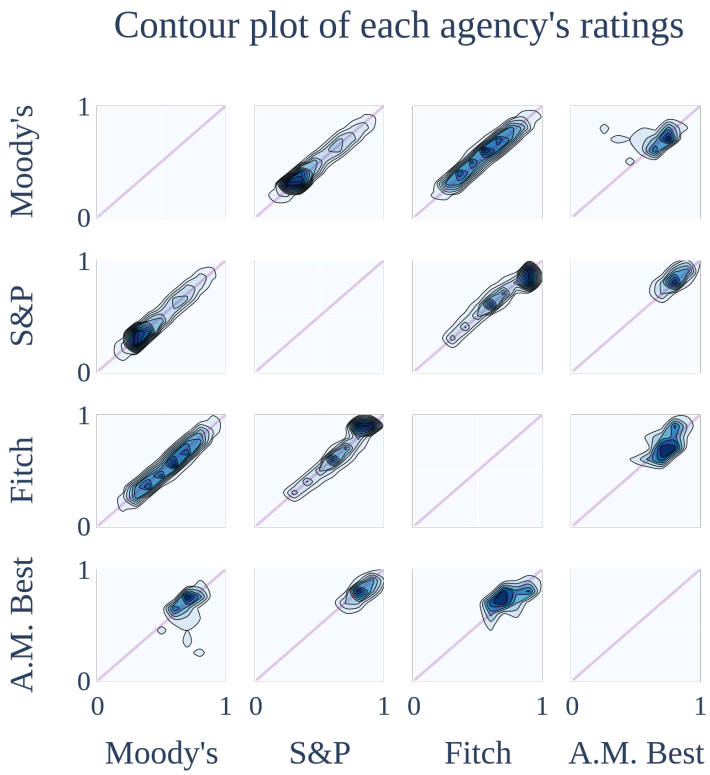

Figure 3.2: Pairwise distribution of the different agencies' ratings. The semi-transparent line contained in each of the distributions reflects an equivalent one-to-one relationship, it does not convey a linear regression of the data. It is just for visual comparison purposes.

This mapping allows us to perform a more in-depth analysis of the structure of the data. For instance, Figure 3.2 allows us to explore the joint distribution of the ratings while Figure 3.3 their correlation for all the different pairs of agencies. Moody's, S&P and Fitch seem to have an equivalent correspondence for their ratings. A.M. Best, shows quite a bit more noise compared to them, which might be caused by the lower number of samples contained in the dataset. This apparent linearity will be leveraged to perform the Transfer Learning phase of the process.

After this, we match the companies between both of our datasets, using the previously discussed keys, to be able to perform the rating prediction. The matching eliminated some companies that lacked observations from either the features or the ratings. As such, our project might have an additional layer of bias created by this artificial selection of companies. It can be seen in Figures 3.4 and 3.5 how the distribution of the ratings changes after the matching. We can observe a general increase in kurtosis, with S&P and Fitch having their modes transferred to a more central point

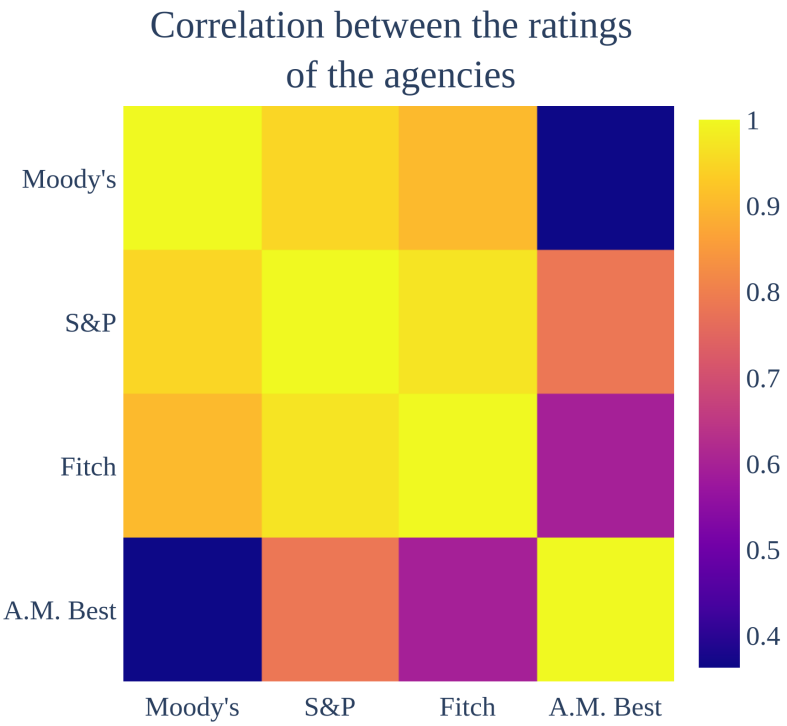

Figure 3.3: Correlation of ratings between agencies.

in the interval.

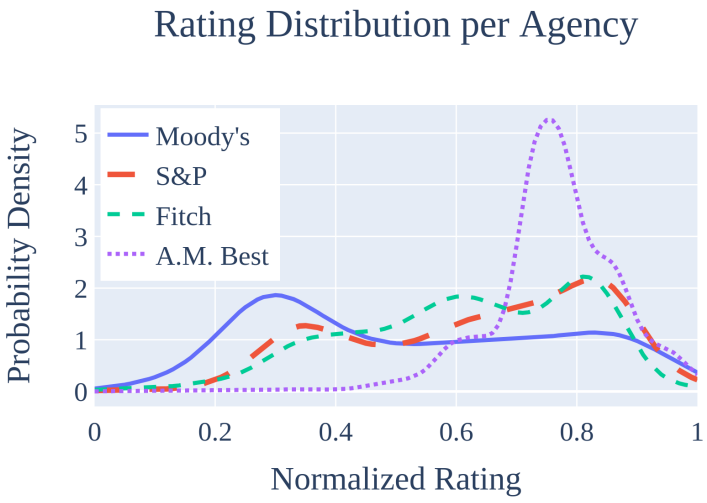

Figure 3.4: Distribution of the ratings per agency before the dataset matching.

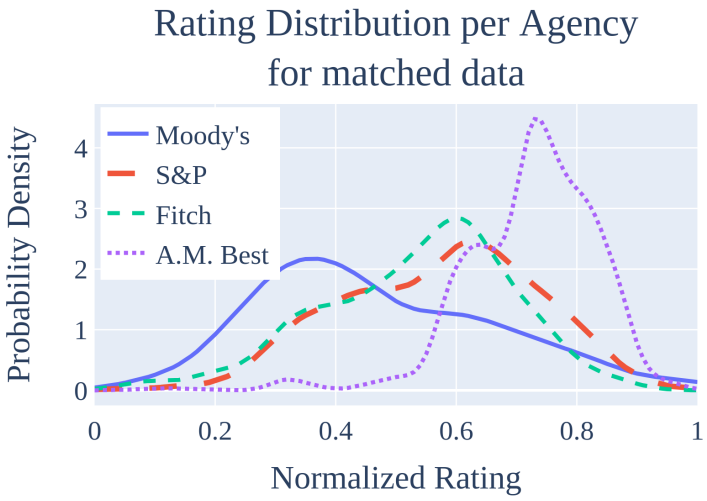

Figure 3.5: Distribution of the ratings per agency after the dataset matching.

Finally, we can study the actual features after the matching. Figure 3.6 shows, in the first plot, the distribution of matched companies by region. Again, the match is heavily skewed to North America, Europe and Asia. This is an artifact of the ratings' dataset, as the features' dataset mainly focused on Japanese companies. This heavily impacted the number of matches retrieved between the data sources. The second plot shows the correlation between the features. There are a lot of areas with strong positive or negative correlation, which is troublesome for a lot of ML models given the noise and unwanted restrictions they add. This further supports the need of dimensionality reduction as introduced in Section 2.1. Also, there are two features (*Long Term Debt*, *Net Assets Per Share*) which contain above of 90% of missing values. These two features will consequently not be considered for further sections of this project.

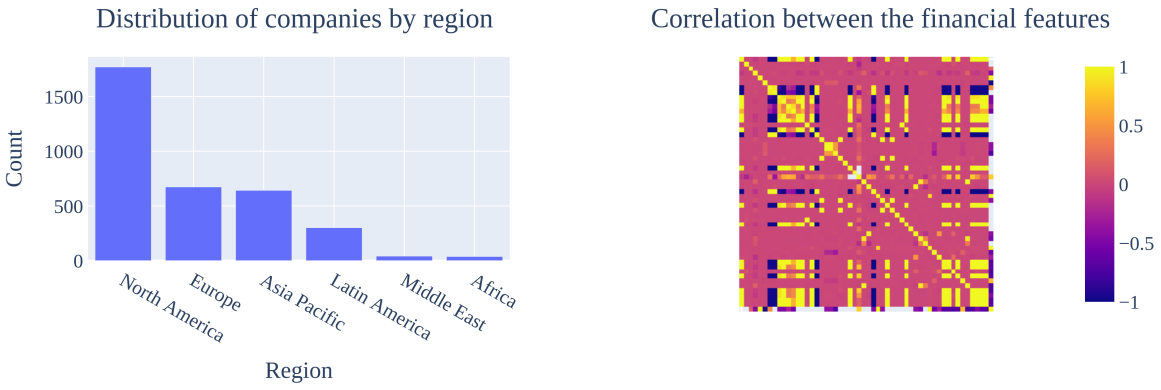

Figure 3.6: Distribution of companies per region (1) and correlation between the features(2).

3.2 Transfer learning paradigm

Recalling from Section 2.3, our problem is of the homogeneous kind, where $\mathcal{X}_S = \mathcal{X}_T$. That is, the financial and business data is the same, as they relate to the same source. Nonetheless, the underlying distributions are not the same. The reason is due to the different specialization of agencies. On top of that, the tasks are completely different, as every agency has their unique labels.

The first step, which is dealing with the different label spaces, was already introduced in Section 3.1.1, where the rating classes were transformed into a numerical domain. As this Section also portrays, marginal and joint distributions over this transformation differ. The proposed method to deal with this issue, as presented in Figure 3.7, is to adapt the labels from the source domains into the target domain, that is, training a mapping $g : \boldsymbol{Y}_S \rightarrow \boldsymbol{y}_T$, where $\boldsymbol{Y}_S = (\boldsymbol{y}_{S_1}, \boldsymbol{y}_{S_2}, \boldsymbol{y}_{S_3})$. These adapted labels will be utilized in two ways: to enlarge the dataset for the feature selection phase, and leveraged as an extra feature in the prediction process to improve accuracy. This will hopefully result in a more robust overall prediction.

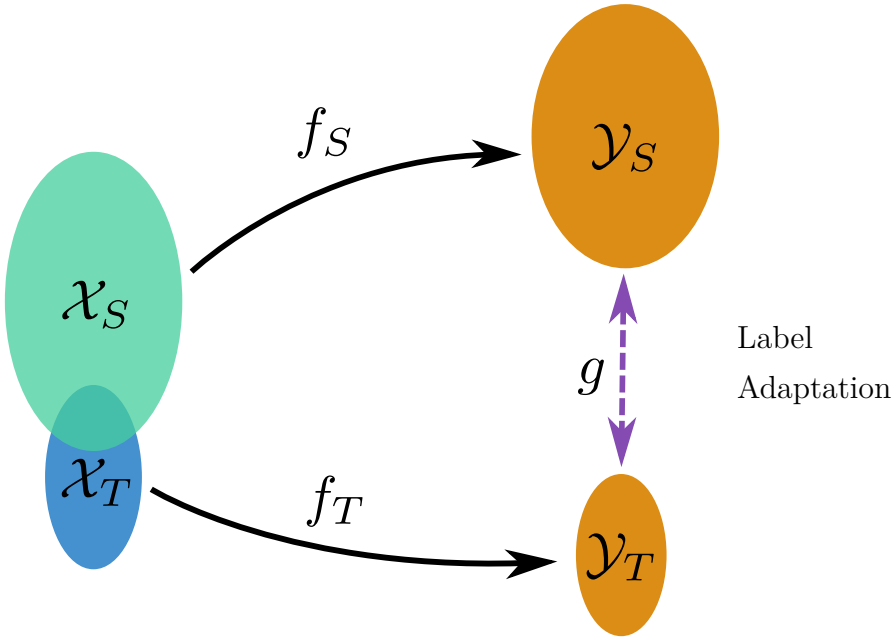

Figure 3.7: Transfer Learning via Label Adaptation.

Given the observed relationships between labels, the algorithm to perform label adaptation is the following:

In Figure 3.8, Ordinary Least Squares (OLS) was applied in Algorithm 1. We can observe that the adapted labels have been skewed to the upper section of the plot, especially in the section of lower ratings. This is probably due to the sensitivity that OLS has to outliers, along with the scarce available data points, which can be easily

Algorithm 1: Label adaptation algorithm

Data: Arrays with the source labels ($\mathcal{Y}_S$) and target labels ($\mathcal{Y}_T$)
Result: Array with adapted target labels ($\bar{\mathcal{Y}}_T$)
initialize *pairs* as the pair combinations of $\mathcal{Y}_S$;
initialize *order* as an empty array;
initialize $\bar{\mathcal{Y}}_S$ as $\mathcal{Y}_S$;
initialize $\bar{\mathcal{Y}}_T$ as $\mathcal{Y}_T$;
initialize j as 0;
sort *pairs* decreasingly by the number of non-empty records;
for *i in 0 to ncols*($\mathcal{Y}_S$) **do**
    append the elements of *pairs*[i] not found in *order* to *order*;
    fill $\bar{\mathcal{Y}}_S[:, order[(j+1):]]$ with a regression on $\bar{\mathcal{Y}}_S[:, order[0:j]]$;
    fill $\bar{\mathcal{Y}}_S[:, order[0:j]]$ with a regression on $\bar{\mathcal{Y}}_S[:, order[(j+1):]]$;
    update j to *length*(*order*);
fill $\bar{\mathcal{Y}}_T$ with a regression on $\bar{\mathcal{Y}}_S$;

noted for Moody's case.

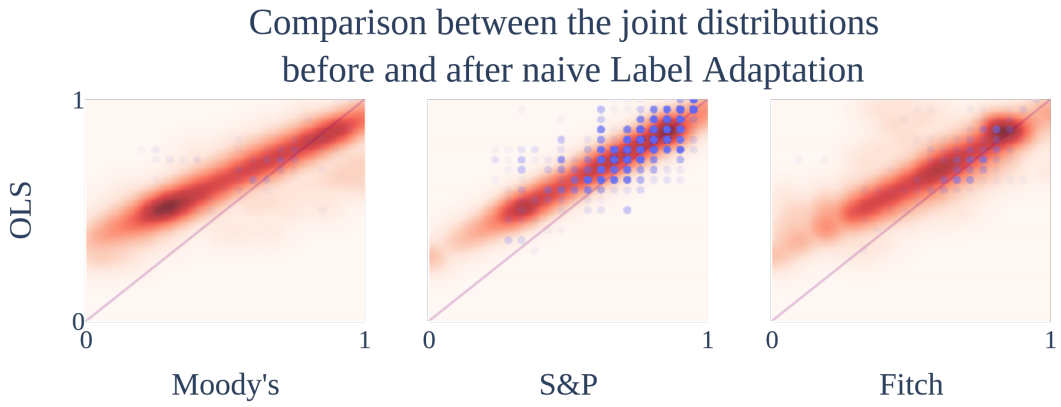

Figure 3.8: Comparison of joint distributions using OLS before (blue dots) and after (red heatmap) naive Label Adaptation from the source to the target labels.

To test the outlier assumption, we perform the following experiment. We regress using OLS with A.M. Best's ratings as our target feature and selecting all the other agencies, individually, as our source features. Then we perform the regression backwards, with A.M. Best as the source and the other agencies as the targets. Figure 3.9 shows how the two regressions differ, even more so for noisy and small data samples. This results can be also observed in Table 3.3, as there is a large gap between the values of the intercepts and the slope coefficients of OLS.

Table 3.3: Intercepts and coefficients for Forward and Backward OLS comparing A.M. Best against the remaining three agencies.

	Intercept		Coefficient	
	Forward OLS	Backward OLS	Forward OLS	Backward OLS
Moody's	0.5895	-0.2282	0.1967	1.4976
S&P	0.2846	-0.0439	0.6757	1.0964
Fitch	0.375	-0.2852	0.5117	1.4488

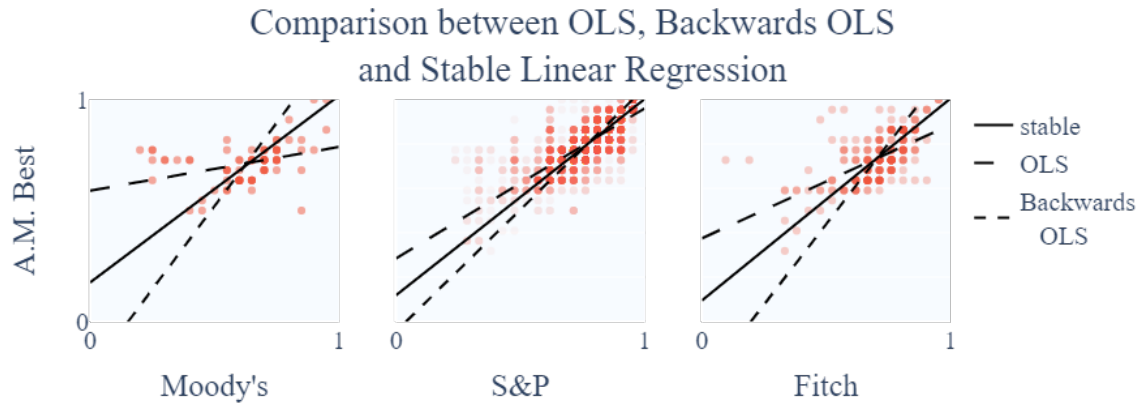

Figure 3.9: Forwards, Backwards and Stable regression on A.M. Best against all other agencies.

The discrepancy in the forward and backward OLS regression poses a problem for Algorithm 1, as it assumes that the direction of the regression does not matter when attempting to fill the ratings database. The differences are caused by the optimization function, as the forward OLS optimizes the quadratic distance on the x-axis, while the backward OLS does so on the y-axis. A new type of regression, induced by the unification of these two optimization functions, is proposed for the current study.

Definition 3.2.1 (Stable Linear Regression)**.** A linear regression problem is said to be stable given

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{U}, \quad \boldsymbol{X} = \boldsymbol{Y}\boldsymbol{\Gamma} + \boldsymbol{V}$$

if and only if $\boldsymbol{\beta} = \boldsymbol{\Gamma}^{+}$, with $\boldsymbol{\Gamma}^{+}$ is the pseudo inverse of $\boldsymbol{\Gamma}$.

Remark. The matrices $\boldsymbol{X} \in \mathbb{R}^{m\times(k+1)}, \boldsymbol{Y} \in \mathbb{R}^{m\times(n+1)}$ most of the time will be assumed to be of the form $\begin{bmatrix} 1 & \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_k \end{bmatrix}$ and $\begin{bmatrix} 1 & \boldsymbol{y}_1 & \cdots & \boldsymbol{y}_n \end{bmatrix}$ respectively. Correspondingly, $\boldsymbol{\beta} \in \mathbb{R}^{(k+1)\times(n+1)}, \Gamma \in \mathbb{R}^{(n+1)\times(k+1)}$ take the forms

$$\begin{bmatrix} 1 & \boldsymbol{\beta}_{\text{intercept}} \\ \mathbf{0} & \boldsymbol{\beta}_{\text{coefficient}} \end{bmatrix} \qquad \begin{bmatrix} 1 & \boldsymbol{\Gamma}_{\text{intercept}} \\ \mathbf{0} & \boldsymbol{\Gamma}_{\text{coefficient}} \end{bmatrix}$$

In particular, the objective function to minimize is $\|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Y}\|^2 + \|\boldsymbol{Y}\boldsymbol{\Gamma} - \boldsymbol{X}\|^2$. In order to solve this, the Particle Swarm Optimization (PSO) algorithm was used to find $\boldsymbol{\beta}$, which was in turn weaved into Algorithm 1 to produce Figure 3.10. After some testing, the optimal values seemed to lay close to the line $\alpha\boldsymbol{\beta} + (1-\alpha)\boldsymbol{\Gamma}^{+}$, and points along this line were chosen as the initial particles to speed up the PSO. However, more research is needed as the solution was not exact. It is apparent that Stable Linear Regression (SLR) does a better job than OLS in extracting the overall trends within the system.

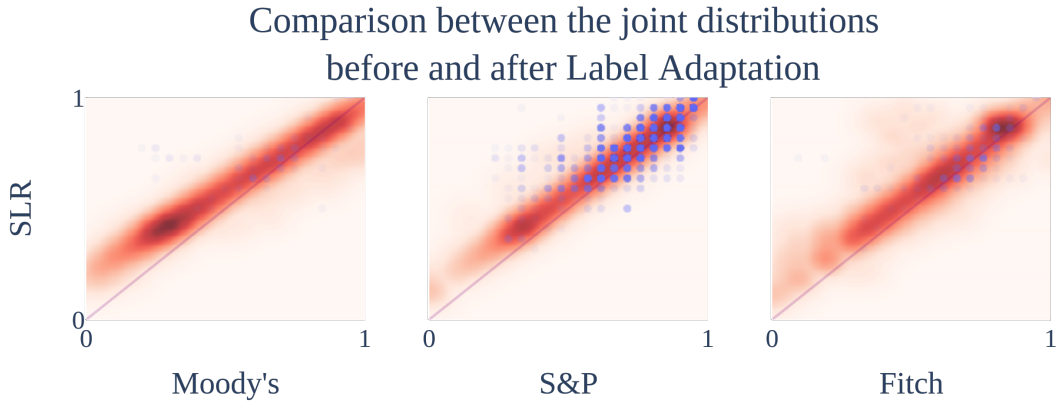

Figure 3.10: Comparison of joint distributions using OLS before (blue dots) and after (red heatmap) stable Label Adaptation from the source to the target labels.

The distribution after performing label adaptation is shown in Figure 3.11. This shows an approximation of the distribution of the ratings if A.M. Best had also rated the same companies as the other agencies. The data now covers a broader interval in the available ratings, which should prove useful for further prediction tasks.

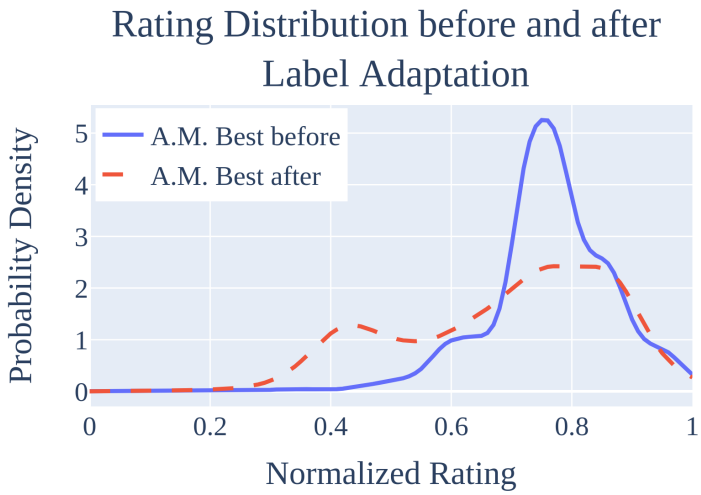

Figure 3.11: Comparison of rating distribution before and after Label Adaptation

The numeric ratings were transformed back to their corresponding classes for further validation. In Figure 3.12 one can observe the confusion matrix of the adapted labels. It is evident the existence of noise in the process, but the overall tendency seems to be preserved. As Figure 3.12 compares adaptations versus actual observations, the bias in the type of companies that A.M. Best rates is evident, which may cause issues down the line when predicting ratings on the low portion of the scale.

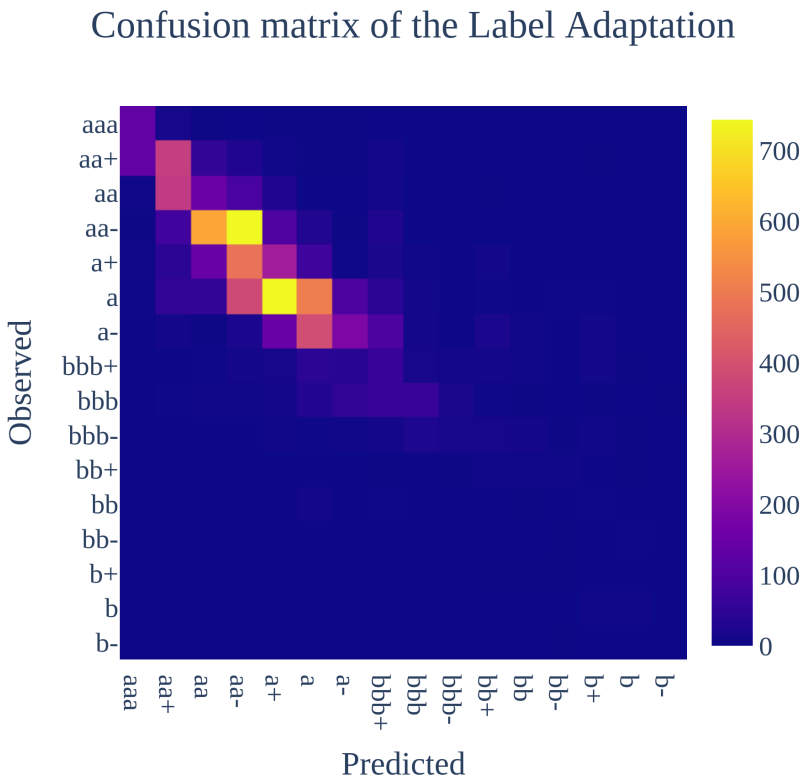

Figure 3.12: Confusion matrix of Label Adaptation after mapping back to the rating scale.

The new "observations" created by the Label Adaptation can be then used to enlarge the dataset for selecting the features. This choice was made because feature selection is based on improvement rather than performance. However, for the actual prediction, the enlarged dataset most certainly will introduce extra noise into the process. To deal with this issue, instead of increasing the number of samples, the adapted labels were incorporated as additional guiding features. This way, we have a simple approximation of what the rating might be, which can be further improved on by the financial features.

3.3 Modelling approaches

An important issue that was not mentioned in 3.1 is the integrity of the financial dataset. It spans various countries (and regulations), which results in missing observations scattered throughout the features. To address this, the following missing imputation process was implemented, given a threshold δ (in this case, $\delta = 0.4$):

1. Normalize the features. To deal with really extreme outliers, we normalized by using the inverse of the cumulative distribution function as fitted by a Gaussian kernel.

2. Check the number of non-missing values per feature, and arrange them in descending order.

3. Repeat the following steps, while cycling through the ordered features, until there is a complete feature sweep without any imputation:

   (a) Select a target feature to be filled according to the order defined previously. This feature will not be dropped in the next step.

   (b) From the dataset containing all the features, drop features iteratively, until the percentage of training samples available (where none of the remaining features are missing) is greater than δ. If this threshold criteria is not met, step (c) is skipped.

   (c) Fill all the the appropriate observations, where the target feature is missing but all the remaining features are non-missing (training dataset), by applying a prediction model. This model does not have to be over complicated, given that the missing values are few and relatively far between.

The imputation was done using OLS for simplicity and efficiency. Only about 10 observations contained missing values after the imputation, resulting in a minimum loss of information.

An obvious decision to be made for the modelling, introduced by the mapping to numerical data, is whether to treat this as a regression or a classification problem. Regression provides a natural way of dealing with ordinality and, at the same time, maintains continuity for the space between each rating step, which may result in additional information. On the other hand, classification can reduce the induced noise of the Label Adaptation dealing a better fit for the prediction. Therefore, both regression and classification will be tested. Specifically, the Multi-Layered Perceptron and Support Vector Models were chosen as a classifier and a regression version exists for both of them, which may provide more insight into this initial choice. The implementation of both models is from [40].

3.3.1 Multi-Layer Perceptron

This model follows the exact same explanation presented in 2.2.1. As a brief reminder for the reader, it works by arranging neurons on a directed acyclical graph. Each neuron's value is computed using the weighted values of the previous layers combined using an activation function. The main difference for the regression and classification versions of MLP is the activation function on the output layer. The regressor uses the identity function ($f(x) = x$), leaving the output unchanged. The classifier can either use the logistic function ($f(x) = 1/\left(1 + e^{-x}\right)$) for binary classifiers, or the Softmax function $\left(f(x)_i = e^{x_i} / \left(\sum_{j=1}^{K} e^{x_j}\right)\right)$ for the multi-class problem, where K is the number of output neurons and x_j is the value of output neuron j. The choice of those functions is not arbitrary as, according to [26], the results can be interpreted as the probabilities of each label.

For both the regressor and the classifier a network structure of 3 hidden layers was selected. Each of the layers is composed of 20 neurons. The structure is kept small, as to avoid over-fitting, given that the experiment is to be carried, by construction, on small datasets. Both applications used the ReLU (stands for Rectified Linear Unit) function ($f(x) = \max(0, x)$) as the activation function for the hidden layers. Furthermore, the classifier used the "ADAM" solver (a spin on the stochastic linear descent) while the regressor used the "LBFGS" solver (a quasi-Newton optimizer).

3.3.2 Support Vector Systems (SVS)

Again, this model is defined as in 2.2.1. The binary case for classification requires a soft margin, and the optimization penalizes for the points that end on the wrong side of the margin. This specific implementation utilizes the one-vs-one heuristic for multi-class problems. This technique partitions the general problem into all possible binary combinations of labels. Then the binary SVM model is applied to each of these combinations, and the label with that is predicted the most is chosen as the final prediction.

Support Vector Regression (SVR) is very similar in nature. As described in [41], the objective function, given a proposed prediction function $f(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{w}) + b$, is

$$\underset{\boldsymbol{w}}{\text{minimize}} \left(K(\boldsymbol{w}, \boldsymbol{w})/2 + C \sum_{i=1}^{N} L(f(\boldsymbol{x}_i), y_i) \right),$$

with a loss function

$$L(f(\boldsymbol{x}), y) = \max(0, |y - f(\boldsymbol{x})| - \epsilon).$$

Here K is the kernel that provides non-linearity, C is the cost of error and ϵ would

be the diameter of the tube were there is no penalization. After some testing, both SVM and SVR used a radial basis function kernel with length scale $1/(d \cdot \text{var}(\text{flat}(\boldsymbol{X})))$ and $C = 1$. SVR used $\epsilon = 0.75$. Here d refers to the number of features and $\text{flat}(\boldsymbol{X})$ is the one dimensional flattened version of $\boldsymbol{X}$. These parameters gave good results and did not seem to affect the results much when modified, but they can be selected more carefully.

3.4 Model validation

Given the relatively small number of samples available for training (around 500 after matching and filtering the companies with actual observed A.M. Best rating), overfitting is a serious concern. To handle this, we implement Cross-Validation (CV). It works by shuffling the observations and partitioning them in k equally-sized* folds. Then, for each of these folds, the model is trained on the remaining data and tested on the fold. The accuracy is then registered as a proxy of how the model performs on unobserved data. To increase the convergence of CV, we perform this task m times, reshuffling the data each repetition, to increase the number of observations and improve convergence to the actual performance. For this case, 10 folds with 10 repetitions were utilized.

With this method, we effectively have 100 comparisons between observations and predictions. We can then group all these $(y_{\text{observed}}, y_{\text{predicted}})$ pairs and plot the extended confusion matrix. This confusion matrix will serve as a visual inspection of how the model captures the overall trend of the data, and inform us of the impact of class imbalance in a qualitative manner.

Finally, we introduce the notion of notch tolerance. The ratings are ordinal in nature and, therefore, we can attribute a sense of distance between the classes. We define the upward and downward steps in the rating as notches (e.g. aaa to aa+ is one notch, aaa to aa is two notches, etc.). We can then define a True Positive as a prediction where the distance of this prediction to the actual observation is lower or equal to a previously specified notch threshold n. The different models were tested for $n \in \{0, 1, 2, 3\}$ and then the accuracy was calculated over these assumptions. The regression models were mapped back to the original labels and then tested using accuracy†.

*Of course, most of the time the last fold will be a bit smaller than the rest.

†Keeping in mind that the actual use case of this study requires the original ratings.

3.5 Feature selection

Given the importance of interpretability in finance, feature selection is selected over feature projection for this project. The relatively low number of features, alongside the reduced number of observations allows us to directly implement a wrapper. However, the implementation of said wrapper will differ from the modelling described in Section 3.3, as here the adapted labels will be used to integrate a bigger training dataset instead of adding just one extra feature. The training dataset is expanded from around 500 observations up to the neighborhood of $9,000$. As mentioned before, we can do this for feature selection as our focus is on improvement and not so much on the accuracy.

The genetic algorithm implementation in [42] was the driver for selecting the features of the different models. This implementation is based on the algorithm described in [43], which can be explained as follows:

1. Generate a random population, where each individual is a subset of the features.
2. Repeat the following steps until there is no change of optimum for a given number of generations, or the maximum number of generations is reached:
   (a) Measure the fitness of each individual.
   (b) Select randomly, based on fitness, which individuals pass on unchanged to the next generation.
   (c) Select randomly, based on fitness, pairs of individuals to be combined into new proposals for the next generation.
3. Return the best individual from all generations as the result of the algorithm.

Only the regression versions of the models were used with this algorithm. The two main reasons are that the regression converged faster in practice, providing additional time for testing; and that the continuous space of the regression could provide subtle hints for the actual contribution of each feature.

4. Results

In this chapter we present the results of the process for counterparty ratings prediction. Further comments on improvements and possible modifications can be found in Chapter 5, but the present chapter will focus on the method as described in Chapter 3.

First, in Section 4.1, we explain the outcomes for feature selection, comparing the results yielded with and without LA. Section 4.2 does so for the actual predictions using accuracy and confusion matrices, with the additional comparison of the classification and regression alternatives.

4.1 Feature Selection Results

We applied the Genetic Algorithm as illustrated in Section 3.5 for each combination of model family (MLP and SVS) with and without the use of LA. We remind the reader that, due to the faster convergence times, only the regression alternatives were utilized, as some smaller scale testing yielded similar results between classification and regression. These results are presented in Tables 4.1 - 4.4:

Table 4.1: Selected features for MLP without Label Adaptation.

Cash and Equivalents
Weighted Average Shares Outstanding - Diluted
Total: 2 features

Table 4.2: Selected features for SVS without Label Adaptation.

Accounts Payable Turnover	EPS - Net Income - Diluted	PP&E Turnover
Calculated Tax Rate	Operating Income	Total Assets Per Share
Total: 6 features		

Table 4.3: Selected features for MLP with Label Adaptation.

Accounts Payable	EPS - Net Income - Basic	Return on Equity
Accounts Payable Turnover	EPS - Net Income - Diluted	Return on Investment
Calculated Tax Rate	Free Cash Flow Per Share	Shares Outstanding
Cash Flow Per Share	Net Current Assets	Total Asset Turnover
Cash, Beginning of Year	Net Income	Total Assets
Current Assets	Net Margin	Total Debt to Equity
EBITDA	Operating Margin	Total Revenue
EBITDA Margin	Return on Assets	Weighted Average Shares Outstanding - Basic
Total: 24 features		

Table 4.4: Selected features for SVS with Label Adaptation.

Accounts Payable	Net Income	Revenue Per Employee
Accounts Payable Turnover	Net Income Per Employee	Total Assets
Calculated Tax Rate	Net Margin	Total Assets Per Share
Cash From Operations	Operating Income	Total Debt to Equity
Cash, Beginning of Year	Operating Margin	Total Liabilities
Cash, End of Year	Receivables Turnover	Total Revenue
EPS - Net Income - Diluted	Return on Assets	Weighted Average Shares Outstanding - Basic
Gross Margin	Return on Equity	Weighted Average Shares Outstanding - Diluted
Total: 24 features		

We notice from Tables 4.1 - 4.4 that the algorithm selected significantly fewer features when no Label Adaptation was applied. This is due to the lower number of samples available for training, which effectively over-fits the model. This exact issue is the main reason behind the difference in methodology between the prediction and the feature selection phase of the project. The features selected are supposed to correct possible errors and generalize for out-of-sample predictions, which is not possible in the structure provided by the model setting*.

*Using the new labels as a feature rather than as a way of expanding the dataset.

4.2 Prediction Results

Table 4.5 gathers the mean accuracies for MLP and SVS without implementing Label Adaptation. Table 4.6 gathers the mean accuracies for MLP and SVS after implementing Label Adaptation. Figure 4.1 presents a comparison between the predictions with and without Label Adaptation for the different configurations proposed.

Table 4.5: Accuracy means for prediction without Label Adaptation.

	Classification		Regression	
Notches	MLP	SVM	MLP	SVR
0	0.2475	**0.2979**	0.2039	0.2257
1	0.4734	0.5569	0.5167	**0.5682**
2	0.7128	**0.7591**	0.7506	0.7555
3	0.8425	0.8518	0.8635	**0.8741**

Table 4.6: Accuracy means for prediction with Label Adaptation.

	Classification		Regression	
Notches	MLP	SVM	MLP	SVR
0	0.2814	**0.3348**	0.2630	0.2756
1	0.5948	0.5876	0.6460	**0.6927**
2	0.8446	0.8012	0.8339	**0.8822**
3	0.9378	0.9229	0.9149	**0.9568**

It can be observed from Tables 4.5 - 4.6 and Figure 4.1 that classification is better than regression for exact prediction, that is, when the notch threshold is 0. However, regression tends to out-perform classification when increasing tolerance. This might be due to the loss of ordinality in the classification versions of the models. Furthermore, SVS seem to give a better prediction than MLPs most of the time. This might be an artifact of over-fitting or under-fitting the MLPs, as the parameter selection process was in no way rigorous. However, this was not a key focus in this study, and the important observation was that improvement exists with LA.

Changing the focus to TL, it appears to increase accuracy on all notch threshold levels, albeit, for low notch threshold levels the improvement is small. This improvement is more clear on the upper notch threshold levels, where accuracy grows around 0.10.

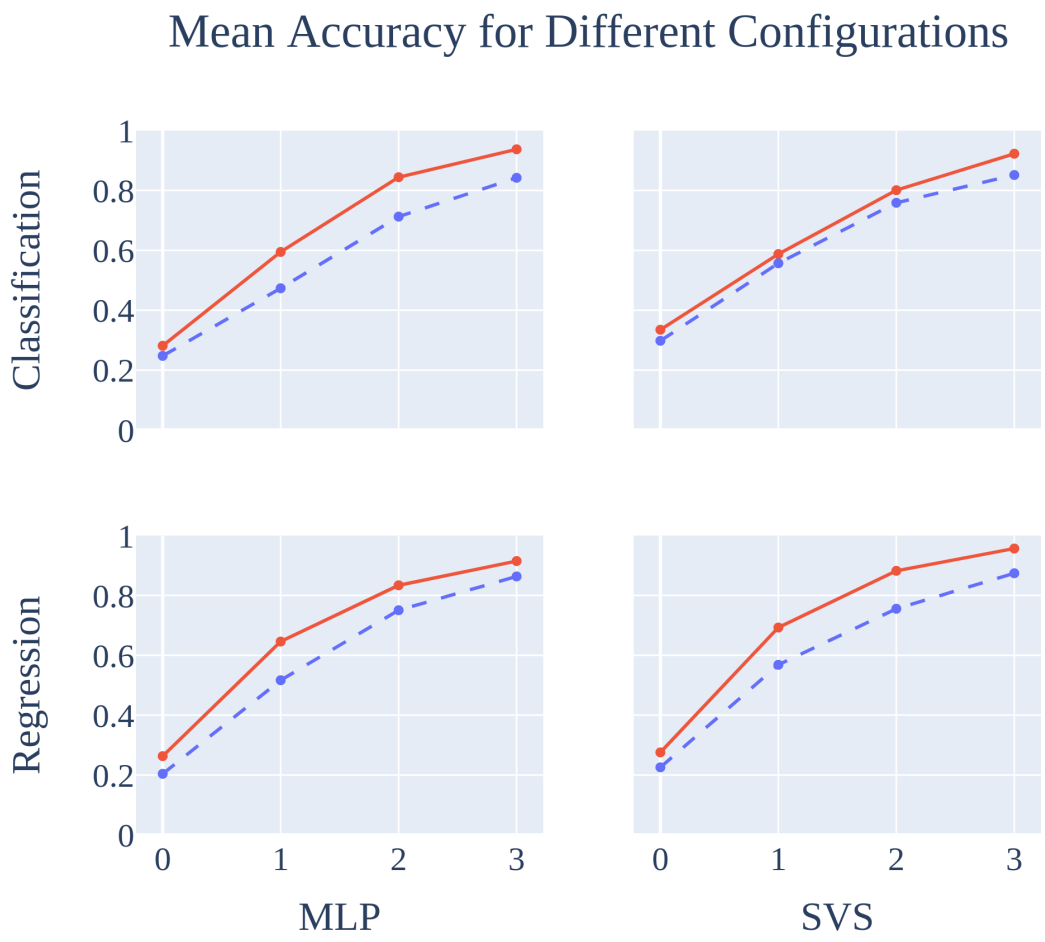

Figure 4.1: Mean Accuracies for the different model configurations. Horizontally we present MLP and SVS variations, while vertically it is the classification and regression alternatives. The x-axis displays the notch threshold while the y-axis refers to the mean accuracy. Predictions using LA are the orange continuous lines and predictions without using LA are the dashed blue lines.

To understand what is really happening with these results, we need to look at Figures 4.2 - 4.5. Label adaptation apparently captures the underlying distribution and trends of the original ratings. All the models that do not use the adapted labels seem to just produce an uncorrelated distribution around the mode of the training data. This suggests a poor performance for out-of-sample prediction, as trying to predict a company with a real rating considerably better or worse than the ones observed in the training set will most likely result in an inaccurate prediction.

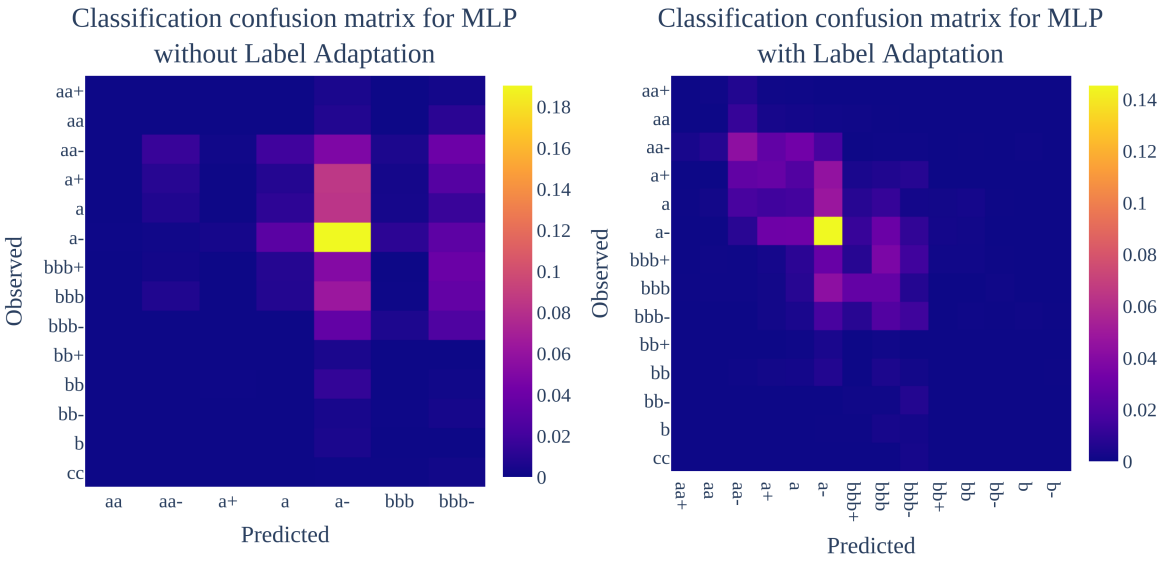

Figure 4.2: Confusion matrix for MLP classifier before (1) and after (2) Label Adaptation.

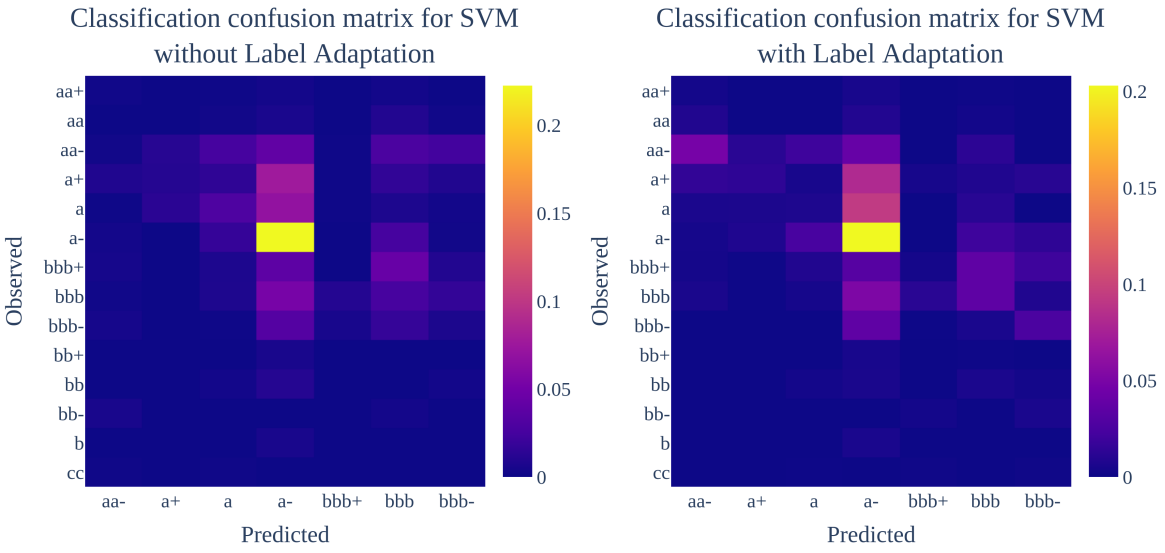

Figure 4.3: Confusion matrix for SVM classifier before (1) and after (2) Label Adaptation.

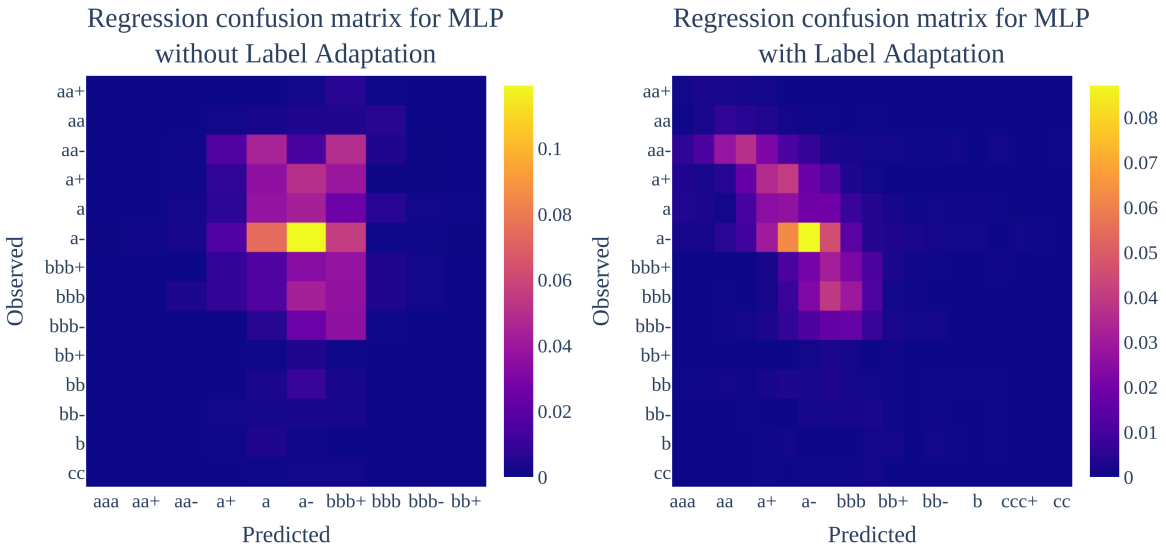

Figure 4.4: Confusion matrix for MLP regressor before (1) and after (2) Label Adaptation.

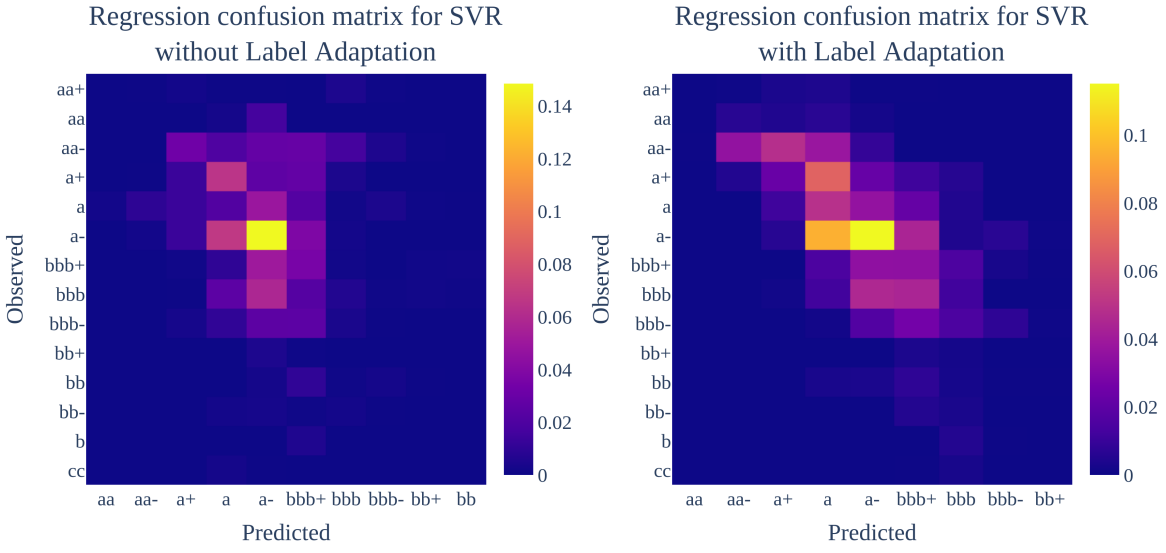

Figure 4.5: Confusion matrix for SVR regressor before (1) and after (2) Label Adaptation.

5. Discussion

Budget was a big constraint for this project, given the price of this kind of data. The financial dataset chosen was good enough for a proof of concept, but a firm considering to implement the process should invest time and resources into making sure the data retrieved fits their own purposes. Specifically, a dataset which maximizes the number of matched ratings is important to prevent loss of information.

Depending on this choice of source for the financial dataset, a different method for missing value imputation might be preferred. In this case, the choice was almost irrelevant due to the sparsity of those missing values, but having concentrations of missing values column-wise or row-wise can be an issue. A deeper look into this topic is suggested, as it was out of the scope of this study.

Stable Linear Regression was introduced as a way to deal with feature systems where forwards and backwards conversions are essential, therefore, as the name suggests, increasing the stability of the system. The main drawback of the current implementation is the lack of an analytical solution. This works for relatively small datasets as some empirical testing uncovered useful heuristics for the solutions, but scalability will be an issue until an analytical form is developed.

On this same line of scalability, a bigger dataset will require a more complex method for feature selection. The literature reviewed in Section 2.1.1 might prove useful, and the reader is advised to test these methods and investigate other implementations that are better suited for the use case.

Chapter 4 mentions the improvement that TL provides, specifically in the sense of generalization for out-of-sample predictions. However, more testing is needed to completely verify this, preferably with a dataset that spans the whole label domain. A proposal is to train the model by excluding either the upper or bottom end of the ratings (as a middle section is easily inferred using SLR), and try to predict the exclusions.

Figure 3.12 in Section 3.2, when compared with the results provided in Chapter 4, suggests that Label Adaptation on its own might be a better prediction than both MLPs and SVS. This is further sustained by Table 5.1 and Figure 5.1[†], where accuracies

[†]The difference between Figures 3.12 and 5.1 is that the latter implements CV, while the former is both trained and tested on all observations.

are higher, and variance is lower. Further tests are needed to verify that the additional information in the financial dataset can in fact improve performance.

Table 5.1: Accuracy means and medians for Stable Linear Regression using cross-validation with 10 folds and 10 repetitions.

Notches	Mean	Median
0	0.3394	0.3349
1	0.7987	0.7920
2	0.9330	0.9358
3	0.9640	0.9652

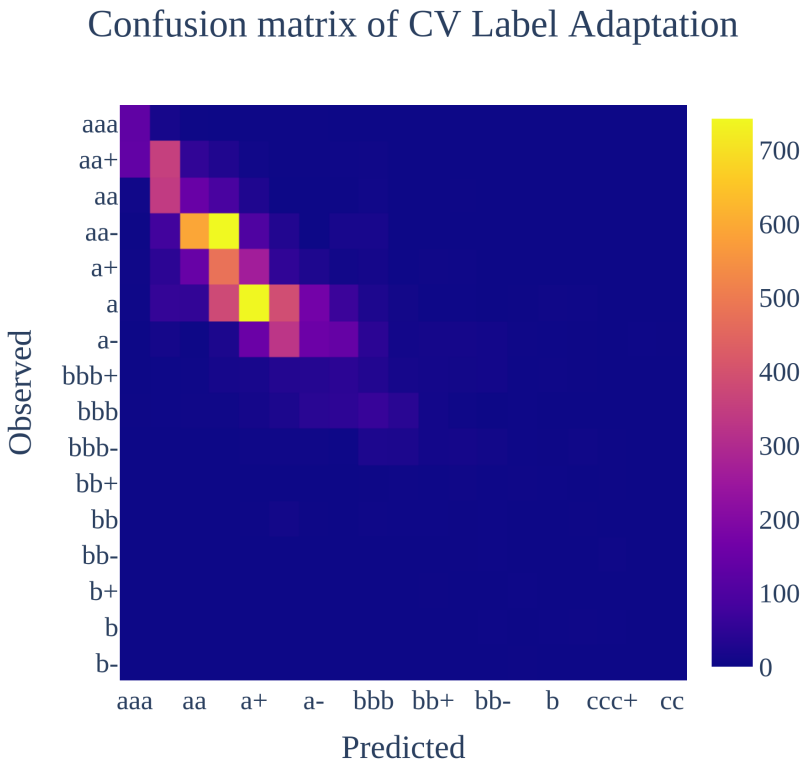

Figure 5.1: Confusion matrix for Stable Linear Regression using cross-validation with 10 folds and 10 repetitions.

6. Conclusions

Credit rating is a key part of risk management. Recognizing the risk originating from a counterparty is easier with ratings, as they provide an intuitive way of comparing the different firms. Current regulations encourage companies operating in the financial sector to develop their own methodologies to perform credit ratings. This thesis presents a new approach for either creating a rating methodology, or testing an existing one via comparison by leveraging Label Adaptation. Label Adaptation extracts the information contained in ratings assigned by expert agencies, and transfers that knowledge into a new rating domain.

Transfer Learning via Label Adaptation improved accuracy on at all different notch threshold levels, being especially better with higher thresholds. Accuracy was enhanced from 0.87 to 0.96 for a 3 notch threshold. These results can be further confirmed qualitatively via the confusion matrices, where TL helped the models match the actual trend of the ratings, instead of just making random guesses distributed near the mode of the data.

More testing is needed, specifically with a dataset spanning a broader set of ratings, as the adapted labels on their own seemed to be a better predictor than the actual predictions. More experiments can be designed to investigate whether the ML models are better at generalizing unobserved sections of the rating scale than SLR. Further model options can be explored, as the reviewed literature provided alternatives specializing in skewed datasets.

Either way, SLR appears to be a viable solution for dealing with systems where the forwards and backwards projections need to produce stable results. However, this model is still in its early stages, and more research is needed to provide an analytic form to be used in considerably larger applications.

Bibliography

[1] D. J. Bolder, *Credit-Risk Modelling: Theoretical Foundations, Diagnostic Tools, Practical Examples, and Numerical Recipes in Python*, 1st ed. Springer International Publishing, 2018.

[2] Basel Committee on Banking Supervision., "Basel iii: A global regulatory framework for more resilient banks and banking systems," 2010. [Online]. Available: https://www.bis.org/publ/bcbs189.pdf

[3] P. Hajek and K. Michalak, "Feature selection in corporate credit rating prediction," *Knowledge-Based Systems*, vol. 51, pp. 72 – 84, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705113002104

[4] H. Suryanto, C. Guan, A. Voumard, and G. Beydoun, "Transfer learning in credit risk," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 483–498.

[5] A. AghaeiRad, N. Cheng, and B. Ribeiro, "Improve credit scoring using transfer of learned knowledge from self-organizing map," *Neural Computing and Applications*, vol. 28, pp. 1329 – 1342, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705113002104

[6] T. Harris, "Credit scoring using the clustered support vector machine," *Expert Systems with Applications*, vol. 42, no. 2, pp. 741–750, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417414005119

[7] Securities and Exchange Commission., "Annual report on nationally recognized statistical rating organizations," 2020. [Online]. Available: https://www.sec.gov/files/2020-annual-report-on-nrsros.pdf

[8] Experian. (2021) Experian's corporate fact sheet. Retrieved on 2021-02-26. [Online]. Available: https://www.experian.com/corporate/experian-corporate-factsheet

[9] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[10] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of big data*, vol. 3, no. 1, pp. 1–40, 2016.

[11] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative review," *J Mach Learn Res*, vol. 10, pp. 66–71, 2009.

[12] Z. Huang, H. Chen, C.-J. Hsu, W.-H. Chen, and S. Wu, "Credit rating analysis with support vector machines and neural networks: a market comparative study," *Decision Support Systems*, vol. 37, no. 4, pp. 543–558, 2004, data mining for financial decision making. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923603000861

[13] K. jae Kim and H. Ahn, "A corporate credit rating model using multi-class support vector machines with an ordinal pairwise partitioning approach," *Computers & Operations Research*, vol. 39, no. 8, pp. 1800–1811, 2012, special Issue: Advances of Operations Research in Service Industry. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054811001936

[14] G. Sermpinis, S. Tsoukas, and P. Zhang, "Modelling market implied ratings using lasso variable selection techniques," *Journal of Empirical Finance*, vol. 48, pp. 19 – 35, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0927539818300318

[15] H. Chen, T. Li, X. Fan, and C. Luo, "Feature selection for imbalanced data based on neighborhood rough sets," *Information Sciences*, vol. 483, pp. 1 – 20, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025519300507

[16] S. Jadhav, H. He, and K. Jenkins, "Information gain directed genetic algorithm wrapper feature selection for credit rating," *Applied Soft Computing*, vol. 69, pp. 541–553, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494618302242

[17] J. Kim, J. Kang, and M. Sohn, "Ensemble learning-based filter-centric hybrid feature selection framework for high-dimensional imbalanced data," *Knowledge-Based Systems*, vol. 220, p. 106901, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121001647

[18] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *Journal of Machine Learning Research*, vol. 16, no. 89, pp. 2859–2900, 2015. [Online]. Available: http://jmlr.org/papers/v16/cunningham15a.html

[19] C. Orsenigo and C. Vercellis, "Linear versus nonlinear dimensionality reduction for banks' credit rating prediction," *Knowledge-Based Systems*, vol. 47, pp. 14–22, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705113000816

[20] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–23, Dec 22 2000, copyright - Copyright American Association for the Advancement of Science Dec 22, 2000; Last updated - 2017-10-31; CODEN - SCIEAS. [Online]. Available: https://search-proquest-com.libproxy.helsinki.fi/scholarly-journals/global-geometric-framework-nonlinear/docview/213581531/se-2?accountid=11365

[21] S.-C. Huang, "Integrating nonlinear graph based dimensionality reduction schemes with svms for credit rating forecasting," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7515–7518, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741740800688X

[22] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1, pp. 1–39, Feb 2010. [Online]. Available: https://link.springer.com/article/10.1007%2Fs10462-009-9124-7#citeas

[23] A. Verikas, Z. Kalsyte, M. Bacauskiene, and A. Gelzinis, "Hybrid and ensemble-based soft computing techniques in bankruptcy prediction: a survey," *Soft Computing*, vol. 14, no. 9, pp. 995–1010, Jul 2010. [Online]. Available: https://doi.org/10.1007/s00500-009-0490-5

[24] I. Bou-Hamad, "Bayesian credit ratings: A random forest alternative approach," *Communications in Statistics - Theory and Methods*, vol. 46, no. 15, pp. 7289–7300, 2017. [Online]. Available: https://doi.org/10.1080/03610926.2016.1148730

[25] J.-P. Li, N. Mirza, B. Rahat, and D. Xiong, "Machine learning and credit ratings prediction in the age of fourth industrial revolution," *Technological Forecasting and Social Change*, vol. 161, p. 120309, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0040162520311355

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[27] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608014002135

[28] T. Kohonen, *Self-organizing maps*, 3rd ed., ser. Springer series in information sciences, 30. Berlin ;: Springer, 2001.

[29] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep 1995. [Online]. Available: https://doi.org/10.1007/BF00994018

[30] C. Grosan and A. Abraham, *Intelligent Systems: A Modern Approach*, ser. Intelligent Systems Reference Library. Springer Berlin Heidelberg, 2011. [Online]. Available: https://books.google.fi/books?id=c1fzgQj5lhkC

[31] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, 1995, pp. 278–282 vol.1.

[32] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani, *Advances in Domain Adaptation Theory : Available Theoretical Results*. San Diego, UNITED KINGDOM: ISTE Press Limited - Elsevier Incorporated, 2019. [Online]. Available: http://ebookcentral.proquest.com/lib/helsinki-ebooks/detail.action?docID=5880599

[33] Quandl. (2021) MF1 mergent global fundamentals data. Retrieved on 2021-02-24. [Online]. Available: https://www.quandl.com/databases/MF1/data

[34] Securities and Exchange Commission., "Final rule: Nationally recognized statistical rating organizations," 2014. [Online]. Available: https://www.sec.gov/ocr/disclosure-of-credit-rating-histories.html

[35] A. Best. (2021) A.M. Best ratings data. Retrieved on 2021-02-24. [Online]. Available: http://www3.ambest.com/ambv/nrsro/ratinghistory.aspx

[36] S. . Poor's. (2021) Standard & Poor's ratings data. Retrieved on 2021-02-09. [Online]. Available: http://www.standardandpoors.com/en_US/web/guest/regulatory/ratingshistory

[37] Moody's. (2021) Moody's ratings data. Retrieved on 2021-02-09. [Online]. Available: https://www.moodys.com/Pages/reg001004.aspx

[38] F. Ratings. (2021) Fitch ratings data. Retrieved on 2021-02-09. [Online]. Available: https://www.fitchratings.com/ratings-history-disclosure

[39] MakoLab USA Inc., "Lei and other identifiers," 2021, retrieved on 2021-02-09. [Online]. Available: https://lei.info/portal/lei-and-other-identifiers/

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[41] P. Lingras and C. Butz, "Rough support vector regression," *European Journal of Operational Research*, vol. 206, no. 2, pp. 445–455, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221709007917

[42] M. Calzolari, "manuel-calzolari/sklearn-genetic: sklearn-genetic 0.4.1," 4 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4661248

[43] J. R. Koza, "Genetic programming as a means for programming computers by natural selection," *Statistics and Computing*, vol. 4, no. 2, pp. 87–112, Jun 1994. [Online]. Available: https://doi.org/10.1007/BF00175355